heraj

2023 09 04

User Documentation

1 Introduction **3**
- 1.1 Install . . . 3
- 1.2 Compatibility . . . 4
- 1.3 Annotations . . . 4

2 Model **5**
- 2.1 AergoKey . . . 5
- 2.2 Encoding . . . 7
- 2.3 Transaction . . . 8
- 2.4 ContractDefinition . . . 12
- 2.5 ContractInvocation . . . 13
- 2.6 EventFilter . . . 14

3 Aergo Client **17**
- 3.1 Build . . . 17
- 3.2 NonceProvider . . . 19
- 3.3 AccountOperation . . . 21
- 3.4 BlockOperation . . . 24
- 3.5 BlockchainOperation . . . 26
- 3.6 TrasactionOperation . . . 27
- 3.7 ContractOperation . . . 29

4 Wallet **33**
- 4.1 KeyStore . . . 33
- 4.2 Wallet Api . . . 36
- 4.3 Transaction Api . . . 38
- 4.4 Query Api . . . 43

5 Contract **53**
- 5.1 Contract Api . . . 53

6 Examples **59**

Heraj heraj Java . Hera Argo . Aergo project client hera . heraj, herajs, herapy . client api .

CHAPTER 1

Introduction

Heraj aergo java framework.

1.1 Install

maven gradle .

1.1.1 Maven

```
<repositories>
  <repository>
    <id>jcenter</id>
    <url>https://jcenter.bintray.com</url>
  </repository>
</repositories>

...

<dependencies>
  <dependency>
    <groupId>io.aergo</groupId>
    <artifactId>heraj-transport</artifactId>
    <version>${herajVersion}</version>
  </dependency>
  <dependency>
    <groupId>io.aergo</groupId>
    <artifactId>heraj-wallet</artifactId>
    <version>${herajVersion}</version>
  </dependency>
  <dependency>
    <groupId>io.aergo</groupId>
```

(continues on next page)

( )

```
    <artifactId>heraj-smart-contract</artifactId>
    <version>${herajVersion}</version>
  </dependency>
</dependencies>
```

1.1.2 Gradle

```
repositories {
  jcenter()
}

...

dependencies {
  implementation "io.aergo:heraj-transport:${herajVersion}"
  implementation "io.aergo:heraj-wallet:${herajVersion}"
  implementation "io.aergo:heraj-smart-contract:${herajVersion}"
}
```

1.2 Compatibility

Heraj jdk7 . jdk8 .

heraj	aergo	jdk	android
1.4.x	2.2.x	7 or higher	3.0 (API 11) or higher
1.3.x	2.0.x, 2.1.x	7 or higher	3.0 (API 11) or higher
1.2.2	1.3.x	7 or higher	3.0 (API 11) or higher

1.3 Annotations

4 annotation .

- @ApiAudience.Public : Heraj api . .
- @ApiAudience.Private : Heraj api . .
- @ApiStability.Stable : Major api.
- @ApiStability.Unstable : api.

Heraj annotation .

CHAPTER 2

Model

2.1 AergoKey

AergoKey .

2.1.1 New

AergoKeyGenerator .

```
AergoKey aergoKey = new AergoKeyGenerator().create();
System.out.println(aergoKey);
```

2.1.2 Export

.

Wallet import format .

```
AergoKey aergoKey = new AergoKeyGenerator().create();
EncryptedPrivateKey wif = aergoKey.exportAsWif("password");
System.out.println("Wallet Import Format: " + wif);
```

Key format .

```
AergoKey aergoKey = new AergoKeyGenerator().create();
KeyFormat keyFormat = aergoKey.exportAsKeyFormat("password");
System.out.println("KeyFormat: " + keyFormat);
```

2.1.3 Import

.

Wallet import format .

```
EncryptedPrivateKey importedWif = EncryptedPrivateKey
    .of("47btMyQmmWddJmEigUp8HjUPam94Jjtf6eG6SW74r61YmbcJGyoxhwTBa8XhVBQ9wYm468DED");
AergoKey imported = AergoKey.of(importedWif, "password");
System.out.println("Imported from wif: " + imported);
```

Key format .

```
String keystore = loadResource(
    "/AmPo7xZJoKNfZXg4NMt9n2saXpKRSkMXwEzqEAfzbVWC71HQL3hn__keystore.txt");
KeyFormat keyFormat = KeyFormat.of(BytesValue.of(keystore.getBytes()));
AergoKey imported = AergoKey.of(keyFormat, "password");
System.out.println("Imported from keyformat: " + imported);
```

2.1.4 Sign and Verify

transaction message . Heraj .

transaction .

```
// prepare aergo key
AergoKey aergoKey = new AergoKeyGenerator().create();

// sign transaction
RawTransaction rawTransaction = RawTransaction.newBuilder(ChainIdHash.of(BytesValue.
↪EMPTY))
    .from(aergoKey.getAddress())
    .to(aergoKey.getAddress())
    .amount(Aer.AERGO_ONE)
    .nonce(1L)
    .build();
Transaction transaction = aergoKey.sign(rawTransaction);
System.out.println("Signed transaction: " + transaction);

// verify transaction
Verifier verifier = new AergoSignVerifier();
boolean result = verifier.verify(transaction);
System.out.println("Verify result: " + result);
```

Plain message . plain message hash .

```
// prepare aergo key
AergoKey aergoKey = new AergoKeyGenerator().create();

// sign message
BytesValue plainMessage = BytesValue.of("test".getBytes());
Signature signature = aergoKey.signMessage(plainMessage);
System.out.println("Signature: " + signature);

// verify signature
Verifier verifier = new AergoSignVerifier();
boolean result = verifier.verify(aergoKey.getAddress(), plainMessage, signature);
System.out.println("Verify result: " + result);
```

Hashed message . hashing .

```
// prepare aergo key
AergoKey aergoKey = new AergoKeyGenerator().create();

// sign sha-256 hashed message
BytesValue plainMessage = BytesValue.of("test".getBytes());
MessageDigest messageDigest = MessageDigest.getInstance("SHA-256");
byte[] rawHashed = messageDigest.digest(plainMessage.getValue());
Hash hashedMessage = Hash.of(BytesValue.of(rawHashed));
Signature signature = aergoKey.signMessage(hashedMessage);
System.out.println("Signature: " + signature);

// verify signature
Verifier verifier = new AergoSignVerifier();
boolean result = verifier.verify(aergoKey.getAddress(), hashedMessage, signature);
System.out.println("Verify result: " + result);
```

2.2 Encoding

Heraj BytesValue encoding/decoding . .

- Hex
- Base58
- Base58 with Checksum
- Base64

2.2.1 Encode

Hex encoding.

```
BytesValue bytesValue = BytesValue.of("test".getBytes());
String encoded = bytesValue.getEncoded(Encoder.Hex);
System.out.println(encoded)
```

Base58 encoding.

```
BytesValue bytesValue = BytesValue.of("test".getBytes());
String encoded = bytesValue.getEncoded(Encoder.Base58);
System.out.println(encoded);
```

Base58 with Checksum encoding.

```
BytesValue bytesValue = BytesValue.of("test".getBytes());
String encoded = bytesValue.getEncoded(Encoder.Base58Check);
System.out.println(encoded);
```

Base64 encoding.

```
BytesValue bytesValue = BytesValue.of("test".getBytes());
String encoded = bytesValue.getEncoded(Encoder.Base64);
System.out.println(encoded);
```

2.2.2 Decode

Hex decoding.

```
String encoded = "74657374";
BytesValue bytesValue = BytesValue.of(encoded, Decoder.Hex);
System.out.println(bytesValue);
```

Base58 decoding.

```
String encoded = "3yZe7d";
BytesValue bytesValue = BytesValue.of(encoded, Decoder.Base58);
System.out.println(bytesValue);
```

Base58 with Checksum decoding.

```
String encoded = "LUC1eAJa5jW";
BytesValue bytesValue = BytesValue.of(encoded, Decoder.Base58Check);
System.out.println(bytesValue);
```

Base64 decoding.

```
String encoded = "dGVzdA==";
BytesValue bytesValue = BytesValue.of(encoded, Decoder.Base64);
System.out.println(bytesValue);
```

2.2.3 Example

Base64 .

```
String encoded =
↪"MEUCIQDP3ywVXX1DP42nTgM6cF95GFfpoEcl4D9ZP+MHO7SgoQIgdq2UiEiSp23lcPFzCHtDmh7pVzsow5x1s8p5Kz0aN7I=
↪";
BytesValue rawSignature = BytesValue.of(encoded, Decoder.Base64);
Signature signature = Signature.of(rawSignature);
System.out.println(signature);
```

2.3 Transaction

Transaction blockchain . transaction .

2.3.1 Make a transaction

Heraj aergo transaction dsl .

Plain Transaction

```
// make a plain transaction
AergoKey aergoKey = new AergoKeyGenerator().create();
ChainIdHash chainIdHash = ChainIdHash.of("6YCMGJu3UN66ULzUuS5R7GTxXLDsSjRdjWPB94EiqMJc
↪");
```

(continues on next page)

( )

```
RawTransaction rawTransaction = RawTransaction.newBuilder()
    .chainIdHash(chainIdHash)
    .from(aergoKey.getAddress())
    .to(aergoKey.getAddress())
    .amount(Aer.AERGO_ONE)
    .nonce(1L)
    .fee(Fee.ZERO)
    .payload(BytesValue.of("contract_payload".getBytes()))
    .build();
Transaction transaction = aergoKey.sign(rawTransaction);
System.out.println("Plain transaction: " + transaction);
```

Deploy Contract Transaction

```
// make a contract definition
ContractDefinition definition = ContractDefinition.newBuilder()
    .encodedContract(

↪"FppTEQaroys1N4P8RcAYYiEhHaQaRE9fzANUx4q2RHDXaRo6TYiTa61n25JcV19grEhpg8qdCWVdsDE2yVfuTKxxcdsTQA2B5z
↪")
    .amount(Aer.ZERO)
    .constructorArgs(1, 2)
    .build();

// make a contract deployment transaction
AergoKey aergoKey = new AergoKeyGenerator().create();
ChainIdHash chainIdHash = ChainIdHash.of("6YCMGJu3UN66ULzUuS5R7GTxXLDsSjRdjWPB94EiqMJc
↪");
RawTransaction rawTransaction = RawTransaction.newDeployContractBuilder()
    .chainIdHash(chainIdHash)
    .from(aergoKey.getAddress())
    .definition(definition)
    .nonce(1L)
    .fee(Fee.ZERO)
    .build();
Transaction transaction = aergoKey.sign(rawTransaction);
System.out.println("Contract deployment transaction: " + transaction);
```

Invoke Contract Transaction

```
// make a contract invocation
ContractInterface contractInterface = dummyContractInterface();
ContractInvocation invocation = contractInterface.newInvocationBuilder()
    .function("set")
    .args("key", "123")
    .delegateFee(false)
    .build();

// make a contract invocation transaction
AergoKey aergoKey = new AergoKeyGenerator().create();
ChainIdHash chainIdHash = ChainIdHash.of("6YCMGJu3UN66ULzUuS5R7GTxXLDsSjRdjWPB94EiqMJc
↪");
RawTransaction rawTransaction = RawTransaction.newInvokeContractBuilder()
```

(continues on next page)

( )

```
    .chainIdHash(chainIdHash)
    .from(aergoKey.getAddress())
    .invocation(invocation)
    .nonce(1L)
    .fee(Fee.ZERO)
    .build();
Transaction transaction = aergoKey.sign(rawTransaction);
System.out.println("Invoke contract transaction: " + transaction);
```

Redeploy Contract

```
// make an new contract definition
ContractDefinition reDeployTarget = ContractDefinition.newBuilder()
    .encodedContract(

↪"FppTEQaroys1N4P8RcAYYiEhHaQaRE9fzANUx4q2RHDXaRo6TYiTa61n25JcV19grEhpg8qdCWVdsDE2yVfuTKxxcdsTQA2B5z
↪")
    .amount(Aer.ZERO)
    .constructorArgs(1, 2)
    .build();

// make a contract redeployment transaction
AergoKey aergoKey = new AergoKeyGenerator().create();
ChainIdHash chainIdHash = ChainIdHash.of("6YCMGJu3UN66ULzUuS5R7GTxXLDsSjRdjWPB94EiqMJc
↪");
RawTransaction rawTransaction = RawTransaction.newReDeployContractBuilder()
    .chainIdHash(chainIdHash)
    .creator(aergoKey.getAddress()) // must be creator
    .contractAddress(
        ContractAddress.of("AmJaNDXoPbBRn9XHh9onKbDKuAzj88n5Bzt7KniYA78qUEc5EwBd"))
    .definition(reDeployTarget)
    .nonce(1L)
    .fee(Fee.ZERO)
    .build();
Transaction transaction = aergoKey.sign(rawTransaction);
System.out.println("Contarct redeployment transaction: " + transaction);
```

Create Name

```
// make an name creation transaction
AergoKey aergoKey = new AergoKeyGenerator().create();
ChainIdHash chainIdHash = ChainIdHash.of("6YCMGJu3UN66ULzUuS5R7GTxXLDsSjRdjWPB94EiqMJc
↪");
RawTransaction rawTransaction = RawTransaction.newCreateNameTxBuilder()
    .chainIdHash(chainIdHash)
    .from(aergoKey.getAddress())
    .name(Name.of("namenamename"))
    .nonce(1L)
    .build();
Transaction transaction = aergoKey.sign(rawTransaction);
System.out.println("Create name transaction: " + transaction);
```

Update Name

```
// make an name update transaction
AergoKey aergoKey = new AergoKeyGenerator().create();
ChainIdHash chainIdHash = ChainIdHash.of("6YCMGJu3UN66ULzUuS5R7GTxXLDsSjRdjWPB94EiqMJc
↪");
RawTransaction rawTransaction = RawTransaction.newUpdateNameTxBuilder()
    .chainIdHash(chainIdHash)
    .from(aergoKey.getAddress())
    .name("namenamename")
    .nextOwner(AccountAddress.of("AmgVbUZiReUVFXdYb4UVMru4ZqyicSsFPqumRx8LfwMKLFk66SNw
↪"))
    .nonce(1L)
    .build();
Transaction transaction = aergoKey.sign(rawTransaction);
System.out.println("Update name transaction: " + transaction);
```

Stake

```
// make a stake transaction
AergoKey aergoKey = new AergoKeyGenerator().create();
ChainIdHash chainIdHash = ChainIdHash.of("6YCMGJu3UN66ULzUuS5R7GTxXLDsSjRdjWPB94EiqMJc
↪");
RawTransaction rawTransaction = RawTransaction.newStakeTxBuilder()
    .chainIdHash(chainIdHash)
    .from(aergoKey.getAddress())
    .amount(Aer.of("10000", Unit.AERGO))
    .nonce(1L)
    .build();
Transaction transaction = aergoKey.sign(rawTransaction);
System.out.println("Stake transaction: " + transaction);
```

Unstake

```
// make a unstake transaction
AergoKey aergoKey = new AergoKeyGenerator().create();
ChainIdHash chainIdHash = ChainIdHash.of("6YCMGJu3UN66ULzUuS5R7GTxXLDsSjRdjWPB94EiqMJc
↪");
RawTransaction rawTransaction = RawTransaction.newUnstakeTxBuilder()
    .chainIdHash(chainIdHash)
    .from(aergoKey.getAddress())
    .amount(Aer.of("10000", Unit.AERGO))
    .nonce(1L)
    .build();
Transaction transaction = aergoKey.sign(rawTransaction);
System.out.println("Unstake transaction: " + transaction);
```

Vote

```
// make a vote transaction
AergoKey aergoKey = new AergoKeyGenerator().create();
ChainIdHash chainIdHash = ChainIdHash.of("6YCMGJu3UN66ULzUuS5R7GTxXLDsSjRdjWPB94EiqMJc
↪");
```

(continues on next page)

( )

```
RawTransaction rawTransaction = RawTransaction.newVoteTxBuilder()
    .chainIdHash(chainIdHash)
    .from(aergoKey.getAddress())
    .voteId("voteBP")
    .candidates(asList("123", "456"))
    .nonce(1L)
    .build();
Transaction transaction = aergoKey.sign(rawTransaction);
System.out.println("Vote transaction: " + transaction);
```

2.3.2 Parse Payload to Model

Heraj transaction payload parsing . ContractInvocation .

Contract Invocation

```
// make a contract invocation
ContractInterface contractInterface = dummyContractInterface();
ContractInvocation invocation = contractInterface.newInvocationBuilder()
    .function("set")
    .args("key", "123")
    .delegateFee(false)
    .build();

// make a contract invocation transaction
AergoKey aergoKey = new AergoKeyGenerator().create();
ChainIdHash chainIdHash = ChainIdHash.of("6YCMGJu3UN66ULzUuS5R7GTxXLDsSjRdjWPB94EiqMJc
↪");
RawTransaction rawTransaction = RawTransaction.newInvokeContractBuilder()
    .chainIdHash(chainIdHash)
    .from(aergoKey.getAddress())
    .invocation(invocation)
    .nonce(1L)
    .fee(Fee.ZERO)
    .build();

// parse contract invocation info
PayloadConverter<ContractInvocation> invocationConverter =
    new ContractInvocationPayloadConverter();
ContractInvocation parsedInvocation = invocationConverter
    .parseToModel(rawTransaction.getPayload());
System.out.println("Parsed contract invocation: " + parsedInvocation.getAddress());
```

2.4 ContractDefinition

ContractDefinition lua smart contract model. lua smart contract , Programming Guide .

2.4.1 Make

.

```
// made by aergoluac --payload {some_contract}.lua
String encodedContract = contractPayload;

// make a contract definition
ContractDefinition contractDefinition = ContractDefinition.newBuilder()
    .encodedContract(encodedContract)
    .build();
System.out.println("Contract definition: " + contractDefinition);
```

.

```
// made by aergoluac --payload {some_contract}.lua
String encodedContract = contractPayload;

// make a contract definition
ContractDefinition contractDefinition = ContractDefinition.newBuilder()
    .encodedContract(encodedContract)
    .constructorArgs("key", 123, "test")
    .build();
System.out.println("Contract definition: " + contractDefinition);
```

.

```
// made by aergoluac --payload {some_contract}.lua
String encodedContract = contractPayload;

// make a contract definition
ContractDefinition contractDefinition = ContractDefinition.newBuilder()
    .encodedContract(encodedContract)
    .constructorArgs("key", 123, "test")
    .amount(Aer.AERGO_ONE)
    .build();
System.out.println("Contract definition: " + contractDefinition);
```

2.5 ContractInvocation

ContractInvocation smart contract model. . ContractInvocation ContractInterface .

2.5.1 Make

.

```
// make a contract invocation
ContractInterface contractInterface = contractInterfaceKeep;
ContractInvocation contractInvocation = contractInterface.newInvocationBuilder()
    .function("set")
    .build();
System.out.println("Contract invocation: " + contractInvocation);
```

.

```
// make a contract invocation
ContractInterface contractInterface = contractInterfaceKeep;
```

(continues on next page)

( )

```
ContractInvocation contractInvocation = contractInterface.newInvocationBuilder()
    .function("set")
    .args("key", 333, "test2")
    .build();
System.out.println("Contract invocation: " + contractInvocation);
```

.

```
// make a contract invocation
ContractInterface contractInterface = contractInterfaceKeep;
ContractInvocation contractInvocation = contractInterface.newInvocationBuilder()
    .function("set")
    .args("key", 333, "test2")
    .amount(Aer.AERGO_ONE)
    .build();
System.out.println("Contract invocation: " + contractInvocation);
```

fee delegation .

```
// make a contract invocation
ContractInterface contractInterface = contractInterfaceKeep;
ContractInvocation contractInvocation = contractInterface.newInvocationBuilder()
    .function("set")
    .args("key", 333, "test2")
    .delegateFee(true)
    .build();
System.out.println("Contract invocation: " + contractInvocation);
```

2.6 EventFilter

Aergo smart contract event . Heraj filter .

2.6.1 Make

Block number filter .

```
// set event filter for specific address in block 1 ~ 10
ContractAddress contractAddress = ContractAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
EventFilter eventFilter = EventFilter.newBuilder(contractAddress)
    .fromBlockNumber(1L)
    .toBlockNumber(10L)
    .build();
System.out.println("Event filter: " + eventFilter);
```

block filter .

```
// set event filter for specific address in recent 1000 block
ContractAddress contractAddress = ContractAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
EventFilter eventFilter = EventFilter.newBuilder(contractAddress)
    .eventName("set")
    .recentBlockCount(1000)
```

(continues on next page)

( )

```
    .build();
System.out.println("Event filter: " + eventFilter);
```

Event filter .

```
// set event filter for specific address with name "set" and args "key" in recent␣
↪1000 block
ContractAddress contractAddress = ContractAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
EventFilter eventFilter = EventFilter.newBuilder(contractAddress)
    .eventName("set")
    .args("key")
    .recentBlockCount(1000)
    .build();
System.out.println("Event filter: " + eventFilter);
```

CHAPTER 3

Aergo Client

3.1 Build

Aergo client builder . overridden.

```
AergoClient aergoClient = new AergoClientBuilder()
    .withEndpoint("localhost:7845")
    .withNonBlockingConnect()  // ignored
    .withBlockingConnect()     // applied
    .build();
```

3.1.1 Endpoint

endpoint . localhost:7845 .

```
// connect to 'localhost:7845'
AergoClient aergoClient = new AergoClientBuilder()
    .withEndpoint("localhost:7845")
    .build();
```

3.1.2 Connect Strategy

.

Non-Blocking netty .

```
// connect to 'localhost:7845' with non-blocking connect
AergoClient aergoClient = new AergoClientBuilder()
    .withEndpoint("localhost:7845")
    .withNonBlockingConnect()
    .build();
```

Blocking okhttp .

```
// connect to 'localhost:7845' with blocking connect
AergoClient aergoClient = new AergoClientBuilder()
    .withEndpoint("localhost:7845")
    .withBlockingConnect()
    .build();
```

3.1.3 Connect Type

Plaintext . plaintext .

```
// connect with plain text
AergoClient aergoClient = new AergoClientBuilder()
    .withEndpoint("localhost:7845")
    .withPlainText()
    .build();
```

Tls . client key pkcs8 .

```
// prepare cert files
InputStream serverCert = loadResourceAsStream("/cert/server.crt");
InputStream clientCert = loadResourceAsStream("/cert/client.crt");
InputStream clientKey = loadResourceAsStream("/cert/client.pem"); // must be pkcs8␣
↪format

// connect with plain text
AergoClient aergoClient = new AergoClientBuilder()
    .withEndpoint("localhost:7845")
    .withTransportSecurity("servername", serverCert, clientCert, clientKey)
    .build();
```

3.1.4 Retry

client . .

```
// retry 3 count with a 1000ms interval
AergoClient aergoClient = new AergoClientBuilder()
    .withRetry(3, 1000L, TimeUnit.MILLISECONDS)
    .build();
```

3.1.5 Timeout

client timeout .

```
// set timeout as 5000ms for each request.
AergoClient aergoClient = new AergoClientBuilder()
    .withTimeout(5000L, TimeUnit.MILLISECONDS)
    .build();
```

3.1.6 Close

Aergo client close. Memory leak .

close method close .

```
// create
AergoClient aergoClient = new AergoClientBuilder()
    .withEndpoint("localhost:7845")
    .withBlockingConnect()
    .withTimeout(10000L, TimeUnit.MILLISECONDS)
    .build();

// ... do some operations

// close
aergoClient.close();
```

Java 7 try-with-resources close .

```
// try-with-resources block
try (AergoClient aergoClient = new AergoClientBuilder()
    .withEndpoint("localhost:7845")
    .build()) {

  // ... do some operations
}
```

3.2 NonceProvider

NonceProvider nonce interface. Heraj thread-safe memmory leak SimpleNonceProvider . SimpleNonceProvider .

3.2.1 Create

SimpleNonceProvider .

.

```
// create nonce provider with capacity 100
NonceProvider nonceProvider = new SimpleNonceProvider(100);
```

.

```
// create nonce provider with capacity 1000
NonceProvider nonceProvider = new SimpleNonceProvider();
```

3.2.2 Bind

nonce . .

.

```
AccountAddress accountAddress = AccountAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
NonceProvider nonceProvider = new SimpleNonceProvider();
nonceProvider.bindNonce(accountAddress, 30L);
System.out.println("Binded nonce: " + nonceProvider.getLastUsedNonce(accountAddress));
```

. .

```
AccountAddress accountAddress = AccountAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
AccountState accountState = client.getAccountOperation().getState(accountAddress);
NonceProvider nonceProvider = new SimpleNonceProvider();
System.out.println("Binded nonce: " + nonceProvider.getLastUsedNonce(accountAddress));
```

3.2.3 Use

nonce . thread-safe.

```
AergoKey signer = richKey;
NonceProvider nonceProvider = new SimpleNonceProvider();
long nonce = nonceProvider.incrementAndGetNonce(signer.getAddress());
System.out.println("Next nonce: " + nonce);
```

nonce .

```
AergoKey signer = richKey;
NonceProvider nonceProvider = new SimpleNonceProvider();
long nonce = nonceProvider.getLastUsedNonce(signer.getAddress());
System.out.println("Last used nonce: " + nonce);
```

3.2.4 Example

```
// prepare signer
AergoKey signer = richKey;

// create an nonce provider
AccountState accountState = client.getAccountOperation().getState(signer.
↪getAddress());
NonceProvider nonceProvider = new SimpleNonceProvider();
nonceProvider.bindNonce(accountState);

// print current
long currentNonce = nonceProvider.getLastUsedNonce(signer.getAddress());
System.out.println("Current nonce: " + currentNonce);

// request using thread pool
AccountAddress accountAddress = AccountAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
ExecutorService service = Executors.newCachedThreadPool();
IntStream.range(0, 1000).forEach(i -> {
  service.submit(() -> {
    // get nonce to use
    long nonce = nonceProvider.incrementAndGetNonce(signer.getAddress());
```

(continues on next page)

( )

```
    client.getTransactionOperation().sendTx(signer, accountAddress, Aer.ONE, nonce,
        Fee.INFINITY, BytesValue.EMPTY);
  });
});

// stop the service
service.awaitTermination(3000L, TimeUnit.MILLISECONDS);
service.shutdown();

// should print 1000
long lastUsedNonce = nonceProvider.getLastUsedNonce(signer.getAddress());
System.out.println("Nonce difference: " + (lastUsedNonce - currentNonce));
```

3.3 AccountOperation

.

3.3.1 Get Account State

.

```
AccountAddress accountAddress = AccountAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
AccountState accountState = client.getAccountOperation().getState(accountAddress);
System.out.println("AccountState: " + accountState);
```

3.3.2 Create Name

Transaction .

```
// prepare a signer
AergoKey signer = richKey;

// make a naming transaction
Name name = randomName();
long nonce = nonceProvider.incrementAndGetNonce(signer.getAddress());
TxHash txHash = client.getAccountOperation().createNameTx(signer, name, nonce);
System.out.println("Create name tx hash: " + txHash);
```

3.3.3 Update Name

. .

```
// prepare a signer
AergoKey signer = richKey;

// create an name
Name name = randomName();
long nonce1 = nonceProvider.incrementAndGetNonce(signer.getAddress());
```

(continues on next page)

( )

```
client.getAccountOperation().createNameTx(signer, name, nonce1);

// sleep
Thread.sleep(2000L);

// update an name
AccountAddress nextOwner = AccountAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
long nonce2 = nonceProvider.incrementAndGetNonce(signer.getAddress());
TxHash txHash = client.getAccountOperation().updateNameTx(signer, name, nextOwner,␣
↪nonce2);
System.out.println("Update name tx hash: " + txHash);
```

3.3.4 Get Name Owner

.

.

```
// get name owner at current block
Name name = Name.of("samplename11");
AccountAddress nameOwner = client.getAccountOperation().getNameOwner(name);
System.out.println("Nonce owner: " + nameOwner);
```

.

```
// get name owner at block 3
Name name = Name.of("samplename11");
AccountAddress nameOwner = client.getAccountOperation().getNameOwner(name, 3);
System.out.println("Nonce owner: " + nameOwner);
```

3.3.5 Stake

Aergo staking.

```
// prepare a signer
AergoKey signer = richKey;

// stake 10000 aergo
Aer amount = Aer.of("10000", Unit.AERGO);
long nonce = nonceProvider.incrementAndGetNonce(signer.getAddress());
TxHash txHash = client.getAccountOperation().stakeTx(signer, amount, nonce);
System.out.println("Stake tx hash: " + txHash);
```

3.3.6 Unstake

Staking Aergo UnStaking.

```
// prepare a signer
AergoKey signer = richKey;

// unstake 10000 aergo
```

(continues on next page)

( )

```
Aer amount = Aer.of("10000", Unit.AERGO);
long nonce = nonceProvider.incrementAndGetNonce(signer.getAddress());
TxHash txHash = client.getAccountOperation().unstakeTx(signer, amount, nonce);
System.out.println("Unstake tx hash: " + txHash);
```

3.3.7 Get Stake Info

staking .

```
AccountAddress accountAddress = AccountAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
StakeInfo stakeInfo = client.getAccountOperation().getStakingInfo(accountAddress);
System.out.println("Stake info: " + stakeInfo);
```

3.3.8 Vote

Vote id candidate .

```
// prepare a signer
AergoKey signer = richKey;

// vote to "voteBP"
List<String> candidates = asList(
↪"16Uiu2HAkwWbv8nKx7S6S5NMvUpTLNeXMVCPr3NTnrx6rBPYYiQ4K");
long nonce = nonceProvider.incrementAndGetNonce(signer.getAddress());
TxHash txHash = client.getAccountOperation().voteTx(signer, "voteBp", candidates,␣
↪nonce);
System.out.println("Vote tx hash: " + txHash);
```

3.3.9 Get Vote of Account

.

```
AccountAddress accountAddress = AccountAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
AccountTotalVote voteInfo = client.getAccountOperation().getVotesOf(accountAddress);
System.out.println("Vote info: " + voteInfo);
```

3.3.10 Get Vote Result

Vote id .

```
// get vote result for vote id "voteBP" for top 23 candidates.
List<ElectedCandidate> elected = client.getAccountOperation().listElected("voteBP",␣
↪23);
System.out.println("Elected: " + elected);
```

3.4 BlockOperation

Block .

3.4.1 Get Block Metadata

Block . null .

.

```
BlockHash blockHash = BlockHash.of("DN9TvryaThbJneSpzaXp5ZsS4gE3UMzKfaXC4x8L5qR1");
BlockMetadata blockMetadata = client.getBlockOperation().getBlockMetadata(blockHash);
System.out.println("Block metadata by hash: " + blockMetadata);
```

.

```
long height = 27_066_653L;
BlockMetadata blockMetadata = client.getBlockOperation().getBlockMetadata(height);
System.out.println("Block metadata by height: " + blockMetadata);
```

3.4.2 List Block Metadata

Block . 1000.

.

```
// block metadatas by from hash to previous 100 block
BlockHash blockHash = BlockHash.of("DN9TvryaThbJneSpzaXp5ZsS4gE3UMzKfaXC4x8L5qR1");
List<BlockMetadata> blockMetadatas = client.getBlockOperation()
    .listBlockMetadatas(blockHash, 100);
System.out.println("Block metadatas by hash: " + blockMetadatas);
```

.

```
// block metadatas by from height to previous 100 block
long height = 27_066_653L;
List<BlockMetadata> blockMetadatas = client.getBlockOperation()
    .listBlockMetadatas(height, 100);
System.out.println("Block metadatas by height: " + blockMetadatas);
```

3.4.3 Get Block

Block . null .

.

```
BlockHash blockHash = BlockHash.of("DN9TvryaThbJneSpzaXp5ZsS4gE3UMzKfaXC4x8L5qR1");
Block block = client.getBlockOperation().getBlock(blockHash);
System.out.println("Block by hash: " + block);
```

.

```
long height = 27_066_653L;
Block block = client.getBlockOperation().getBlock(height);
System.out.println("Block by height: " + block);
```

3.4.4 Block Metadata Subscription

Block .

```
// make a subscription
Subscription<BlockMetadata> subscription = client
    .getBlockOperation().subscribeBlockMetadata(new StreamObserver<BlockMetadata>() {
      @Override
      public void onNext(BlockMetadata value) {
        System.out.println("Next block metadata: " + value);
      }

      @Override
      public void onError(Throwable t) {

      }

      @Override
      public void onCompleted() {
      }
    });

// wait for a while
Thread.sleep(2000L);

// unsubscribe it
subscription.unsubscribe();
```

3.4.5 Block Subscription

Block .

```
// make a subscription
Subscription<Block> subscription = client.getBlockOperation()
    .subscribeBlock(new StreamObserver<Block>() {
      @Override
      public void onNext(Block value) {
        System.out.println("Next block: " + value);
      }

      @Override
      public void onError(Throwable t) {
      }

      @Override
      public void onCompleted() {
      }
    });

// wait for a while
```

(continues on next page)

( )

```
Thread.sleep(2000L);

// unsubscribe it
subscription.unsubscribe();
```

3.5 BlockchainOperation

Blockchain Node .

3.5.1 Get Chain Id Hash

node chain id hash .

```
ChainIdHash chainIdHash = client.getBlockchainOperation().getChainIdHash();
System.out.println("Chain id hash: " + chainIdHash);
```

3.5.2 Get Blockchain Status

node blockchain .

```
BlockchainStatus blockchainStatus = client.getBlockchainOperation().
↪getBlockchainStatus();
System.out.println("Blockchain status: " + blockchainStatus);
```

3.5.3 Get Chain Info

node chain .

```
ChainInfo chainInfo = client.getBlockchainOperation().getChainInfo();
System.out.println("Chain info: " + chainInfo);
```

3.5.4 Get Chain Stats

node chain .

```
ChainStats chainStats = client.getBlockchainOperation().getChainStats();
System.out.println("Chain stats: " + chainStats);
```

3.5.5 List Peers

node peer .

hidden peer .

```
List<Peer> peers = client.getBlockchainOperation().listPeers(false, false);
System.out.println("Peers: " + peers);
```

hidden peer .

```
List<Peer> peers = client.getBlockchainOperation().listPeers(true, true);
System.out.println("Peers: " + peers);
```

3.5.6 List Peers Metrics

node peer .

```
List<PeerMetric> peerMetrics = client.getBlockchainOperation().listPeerMetrics();
System.out.println("PeerMetrics: " + peerMetrics);
```

3.5.7 Get Server Info

node . Category .

```
List<String> categories = emptyList();
ServerInfo serverInfo = client.getBlockchainOperation().getServerInfo(categories);
System.out.println("Server info: " + serverInfo);
```

3.5.8 Get Node Status

node .

```
NodeStatus nodeStatus = client.getBlockchainOperation().getNodeStatus();
System.out.println("Node status: " + nodeStatus);
```

3.6 TrasactionOperation

Transaction .

3.6.1 Get Transaction

Transaction . null .

```
TxHash txHash = TxHash.of("39vLyMqsg1mTT9mF5NbADgNB2YUiRVsT6SUkDujBZme8");
Transaction transaction = client.getTransactionOperation().getTransaction(txHash);
System.out.println("Transaction: " + transaction);
```

3.6.2 Get Transaction Receipt

Transaction . null .

```
TxHash txHash = TxHash.of("39vLyMqsg1mTT9mF5NbADgNB2YUiRVsT6SUkDujBZme8");
TxReceipt txReceipt = client.getTransactionOperation().getTxReceipt(txHash);
System.out.println("Transaction receipt: " + txReceipt);
```

3.6.3 Commit

transaction commit. transaction *Transaction*.

```
// get chain id hash
ChainIdHash chainIdHash = client.getBlockchainOperation().getChainIdHash();

// prepare signer
AergoKey signer = richKey;

// make a transaction
long nonce = nonceProvider.incrementAndGetNonce(signer.getAddress());
RawTransaction rawTransaction = RawTransaction.newBuilder()
    .chainIdHash(chainIdHash)
    .from(signer.getAddress())
    .to(signer.getAddress())
    .amount(Aer.AERGO_ONE)
    .nonce(nonce)
    .fee(Fee.ZERO)
    .payload(BytesValue.of("contract_payload".getBytes()))
    .build();

// sign raw transaction
Transaction transaction = signer.sign(rawTransaction);

// commit signed one
TxHash txHash = client.getTransactionOperation().commit(transaction);
System.out.println("Commit tx hash: " + txHash);
```

3.6.4 Send

Aergo .

.

```
// prepare signer
AergoKey signer = richKey;

// make a send transaction
AccountAddress accountAddress = AccountAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
long nonce = nonceProvider.incrementAndGetNonce(signer.getAddress());
TxHash txHash = client.getTransactionOperation()
    .sendTx(signer, accountAddress, Aer.ONE, nonce, Fee.INFINITY, BytesValue.EMPTY);
System.out.println("Send tx hash: " + txHash);
```

.

```
// prepare signer
AergoKey signer = richKey;

// create an name
Name name = randomName();
long nonce1 = nonceProvider.incrementAndGetNonce(signer.getAddress());
client.getAccountOperation().createNameTx(signer, name, nonce1);
```

(continues on next page)

( )

```
// sleep
Thread.sleep(2000L);

// make a send transaction
long nonce2 = nonceProvider.incrementAndGetNonce(signer.getAddress());
TxHash txHash = client.getTransactionOperation()
    .sendTx(signer, name, Aer.ONE, nonce2, Fee.INFINITY, BytesValue.EMPTY);
System.out.println("Send tx hash: " + txHash);
```

3.7 ContractOperation

Provides contract related operations. For more about writing smart contract, see Aergo Smart Contract.

3.7.1 Deploy

Smart contract . .

-> confirm -> contract tx receipt -> contract -> contract interface

Contract definition *ContractDefinition*.

```
AergoKey signer = richKey;

// made by aergoluac --payload {some_contract}.lua
String encodedContract = contractPayload;

// make a contract definition
ContractDefinition contractDefinition = ContractDefinition.newBuilder()
    .encodedContract(encodedContract)
    .build();

// deploy
long nonce = nonceProvider.incrementAndGetNonce(signer.getAddress());
TxHash txHash = client.getContractOperation().deployTx(signer, contractDefinition,
    nonce, Fee.ZERO);
System.out.println("Contract deployment tx hash: " + txHash);

// wait deploy contract to be confirmed
Thread.sleep(2200L);

// get contract tx receipt
ContractTxReceipt contractTxReceipt = client.getContractOperation()
    .getContractTxReceipt(txHash);
System.out.println("Contract tx receipt: " + contractTxReceipt);

// find a contract address
ContractAddress contractAddress = contractTxReceipt.getContractAddress();

// get contract interface
ContractInterface contractInterface = client.getContractOperation()
    .getContractInterface(contractAddress);
System.out.println("Contract interface: " + contractInterface);
```

3.7.2 Re-Deploy

contract . contract logic . private mode aergo . Contract definition *ContractDefinition*.

```
// prepare signer
AergoKey signer = richKey;

// made by aergoluac --payload {some_contract}.lua
String encodedContract = contractPayload;

// make a contract definition
ContractDefinition newDefinition = ContractDefinition.newBuilder()
    .encodedContract(encodedContract)
    .build();

// redeploy
ContractAddress contractAddress = contractAddressKeep;
long nonce = nonceProvider.incrementAndGetNonce(signer.getAddress());
TxHash txHash = client.getContractOperation()
    .redeployTx(signer, contractAddress, newDefinition, nonce, Fee.ZERO);
System.out.println("Redeploy tx hash: " + txHash);
```

3.7.3 Get Contract Tx Receipt

contract tx . null .

```
TxHash txHash = TxHash.of("EGXNDgjY2vQ6uuP3UF3dNXud54dF4FNVY181kaeQ26H9");
ContractTxReceipt contractTxReceipt = client.getContractOperation()
    .getContractTxReceipt(txHash);
System.out.println("ContractTxReceipt: " + contractTxReceipt);
```

3.7.4 Get Contract Interface

contract interface . null .

```
ContractAddress contractAddress = ContractAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
ContractInterface contractInterface = client.getContractOperation()
    .getContractInterface(contractAddress);
System.out.println("ContractInterface: " + contractInterface);
```

3.7.5 Execute

contract . Contract invocation *ContractInvocation*.

```
// prepare signer
AergoKey signer = richKey;

// make a contract invocation
ContractInterface contractInterface = contractInterfaceKeep;
ContractInvocation invocation = contractInterface.newInvocationBuilder()
    .function("set")
    .args("key", 333, "test2")
```

(continues on next page)

( )

```
    .build();

// execute
long nonce = nonceProvider.incrementAndGetNonce(signer.getAddress());
TxHash txHash = client.getContractOperation()
    .executeTx(signer, invocation, nonce, Fee.ZERO);
System.out.println("Execute tx hash: " + txHash);
```

3.7.6 Query

contract . Contract invocation *ContractInvocation*.

```
// java bean
public class Data {

  protected int intVal;

  protected String stringVal;

  public int getIntVal() {
    return intVal;
  }

  public void setIntVal(int intVal) {
    this.intVal = intVal;
  }

  public String getStringVal() {
    return stringVal;
  }

  public void setStringVal(String stringVal) {
    this.stringVal = stringVal;
  }

  @Override
  public String toString() {
    return "Data [intVal=" + intVal + ", stringVal=" + stringVal + "]";
  }

}
```

```
// make a contract invocation
ContractInterface contractInterface = contractInterfaceKeep;
ContractInvocation query = contractInterface.newInvocationBuilder()
    .function("get")
    .args("key")
    .build();

// query contract
ContractResult queryResult = client.getContractOperation().query(query);
Data data = queryResult.bind(Data.class);
System.out.println("Raw contract result: " + queryResult); // { "intVal": 123,
↪"stringVal": "test" }
System.out.println("Binded data: " + data);
```

3.7.7 List Event

block event . Event filter *EventFilter*.

```
ContractAddress contractAddress = contractAddressKeep;
EventFilter eventFilter = EventFilter.newBuilder(contractAddress)
    .eventName("set")
    .args("key")
    .recentBlockCount(1000)
    .build();
List<Event> events = client.getContractOperation().listEvents(eventFilter);
System.out.println("Events: " + events);
```

3.7.8 Event Subscription

event . Event filter *EventFilter*.

```
// prepare signer
AergoKey signer = richKey;

// subscribe event
ContractAddress contractAddress = contractAddressKeep;
EventFilter eventFilter = EventFilter.newBuilder(contractAddress).build();
Subscription<Event> subscription = client.getContractOperation()
    .subscribeEvent(eventFilter, new StreamObserver<Event>() {
      @Override
      public void onNext(Event value) {
        System.out.println("Next event: " + value);
      }

      @Override
      public void onError(Throwable t) {
      }

      @Override
      public void onCompleted() {
      }
    });

// execute
ContractInterface contractInterface = contractInterfaceKeep;
ContractInvocation run = contractInterface.newInvocationBuilder()
    .function("set")
    .args("key", 333, "test2")
    .build();
long nonce = nonceProvider.incrementAndGetNonce(signer.getAddress());
client.getContractOperation().executeTx(signer, run, nonce, Fee.ZERO);
Thread.sleep(2200L);

// unsubscribe event
subscription.unsubscribe();
```

CHAPTER 4

Wallet

4.1 KeyStore

KeyStore AergoKey . KeyStore , , . .

- InMemoryKeystore
- JavaKeyStore
- AergoKeyStore

4.1.1 Create

Heraj KeyStore factory method .

InMemoryKeystore

InMemoryKeystore AergoKey .

```
// make a keystore
KeyStore keyStore = KeyStores.newInMemoryKeyStore();
System.out.println("InMemoryKeystore: " + keyStore);
System.out.println("Stored keys: " + keyStore.listIdentities());
```

JavaKeyStore

JavaKeyStore java.security.keystore . factory method

```
// create a java keystore
java.security.KeyStore delegate = java.security.KeyStore.getInstance("PKCS12");
delegate.load(new FileInputStream(someDir + "/keystore.p12"), "password".
↪toCharArray());
```

(continues on next page)

( )

```
// make a keystore
KeyStore keyStore = KeyStores.newJavaKeyStore(delegate);
System.out.println("JavaKeyStore: " + keyStore);
System.out.println("Stored keys: " + keyStore.listIdentities());
```

AergoKeyStore

AergoKeyStore AergoKey aergo . keystore aergocli keystore . aergocli keystore Creating Accounts.

```
// make a keystore
String root = someDir + "/aergo_keystore";
KeyStore keyStore = KeyStores.newAergoKeyStore(root);
System.out.println("AergoKeyStore: " + keyStore);
System.out.println("Stored keys: " + keyStore.listIdentities());
```

4.1.2 Save and Load

AergoKey .

alias .

```
// create a keystore
KeyStore keyStore = KeyStores.newInMemoryKeyStore();

// create an new key
AergoKey key = new AergoKeyGenerator().create();

// save
Authentication authentication = Authentication.of(KeyAlias.of("myalias"), "password");
keyStore.save(authentication, key);
```

.

```
// create an keystore
KeyStore keyStore = KeyStores.newInMemoryKeyStore();

// create an new key
AergoKey key = new AergoKeyGenerator().create();

// save
Authentication authentication = Authentication.of(key.getAddress(), "password");
keyStore.save(authentication, key);
```

4.1.3 Remove

keystore AergoKey .

alias .

```
// create a keystore
KeyStore keyStore = KeyStores.newInMemoryKeyStore();
```

(continues on next page)

( )

```
// create an new key
AergoKey key = new AergoKeyGenerator().create();

// save
Authentication authentication = Authentication.of(KeyAlias.of("myalias"), "password");
keyStore.save(authentication, key);

// remove
System.out.println("Before remove: " + keyStore.listIdentities());
keyStore.remove(authentication);
System.out.println("After remove: " + keyStore.listIdentities());
```

.

```
// create a keystore
KeyStore keyStore = KeyStores.newInMemoryKeyStore();

// create an new key
AergoKey key = new AergoKeyGenerator().create();

// save
Authentication authentication = Authentication.of(key.getAddress(), "password");
keyStore.save(authentication, key);

// remove
System.out.println("Before remove: " + keyStore.listIdentities());
keyStore.remove(authentication);
System.out.println("After remove: " + keyStore.listIdentities());
```

4.1.4 Export

keystore AergoKey wallet import format export .

alias .

```
// create a keystore
KeyStore keyStore = KeyStores.newInMemoryKeyStore();

// create an new key
AergoKey key = new AergoKeyGenerator().create();

// save
Authentication authentication = Authentication.of(KeyAlias.of("myalias"), "password");
keyStore.save(authentication, key);

// export
EncryptedPrivateKey exported = keyStore.export(authentication, "newpassword");
System.out.println("Exported: " + exported);
```

.

```
// create a keystore
KeyStore keyStore = KeyStores.newInMemoryKeyStore();
```

(continues on next page)

( )

```
// create an new key
AergoKey key = new AergoKeyGenerator().create();

// save
Authentication authentication = Authentication.of(key.getAddress(), "password");
keyStore.save(authentication, key);

// export
EncryptedPrivateKey exported = keyStore.export(authentication, "newpassword");
System.out.println("Exported: " + exported);
```

4.1.5 List Stored Identities

keystore .

```
// create a keystore
KeyStore keyStore = KeyStores.newInMemoryKeyStore();

// create an new key
AergoKey key = new AergoKeyGenerator().create();

// save
Authentication authentication = Authentication.of(KeyAlias.of("myalias"), "password");
keyStore.save(authentication, key);

// list
List<Identity> identities = keyStore.listIdentities();
System.out.println("Stored identities: " + identities);
```

4.1.6 Store

keystore . JavaKeyStore . .

```
// prepare a java keystore
java.security.KeyStore delegate = java.security.KeyStore.getInstance("PKCS12");
delegate.load(null, null);

// create a java keystore
KeyStore keyStore = KeyStores.newJavaKeyStore(delegate);

// store
String path = someDir + "/" + randomUUID().toString();
keyStore.store(path, "password".toCharArray());
```

4.2 Wallet Api

WalletApi *KeyStore* . WalletApi keystore key unlock, lock . WalletApi aergo client high-level api . TransactionApi nonce nonce .

WalletApi unlock . unlock unlock lock.

4.2.1 Create

WalletApi KeyStore .

nonce .

```
// create a keystore
KeyStore keyStore = KeyStores.newInMemoryKeyStore();

// create a wallet api
WalletApi walletApi = new WalletApiFactory().create(keyStore);
System.out.println("WalletApi: " + walletApi);
```

nonce .

```
// create a keystore
KeyStore keyStore = KeyStores.newInMemoryKeyStore();

// create a wallet api with retry count 5 and interval 1s
TryCountAndInterval tryCountAndInterval = TryCountAndInterval
    .of(5, Time.of(1L, TimeUnit.SECONDS));
WalletApi walletApi = new WalletApiFactory().create(keyStore, tryCountAndInterval);
System.out.println("WalletApi: " + walletApi);
```

4.2.2 Unlock and Lock

Unlock transaction .

```
// create a keystore
KeyStore keyStore = KeyStores.newInMemoryKeyStore();

// store new key to keystore
AergoKey aergoKey = new AergoKeyGenerator().create();
Authentication authentication = Authentication.of(aergoKey.getAddress(), "password");
keyStore.save(authentication, aergoKey);

// create a wallet api
WalletApi walletApi = new WalletApiFactory().create(keyStore);

// unlock account
boolean unlockResult = walletApi.unlock(authentication);
System.out.println("Unlock result: " + unlockResult);
System.out.println("Currently locked one: " + walletApi.getPrincipal());

// do something..
Signature signature = walletApi.signMessage(BytesValue.of("test".getBytes()));
System.out.println("Signature: " + signature);

// lock account
boolean lockResult = walletApi.lock();
System.out.println("Lock result: " + lockResult);
```

4.2.3 High Level Api

WalletApi aergo node high-level api . TransactionApi unlock. QueryApi unlock .

```
// prepare client
AergoClient aergoClient = new AergoClientBuilder().build();

// create a keystore
KeyStore keyStore = KeyStores.newInMemoryKeyStore();

// create a wallet api
WalletApi walletApi = new WalletApiFactory().create(keyStore);
System.out.println("WalletApi: " + walletApi);

// transaction api
TransactionApi transactionApi = walletApi.with(aergoClient).transaction();
System.out.println("Transaction Api: " + transactionApi);

// query api
QueryApi queryApi = walletApi.with(aergoClient).query();
System.out.println("Query Api: " + queryApi);
```

4.3 Transaction Api

TransactionApi transaction high-level api . transaction unlock .

4.3.1 Create Name

Unlock .

```
// unlock specific account with authentication
walletApi.unlock(authentication);

Name name = randomName();
TxHash txHash = walletApi.with(client).transaction().createName(name);
System.out.println("Create name tx hash: " + txHash);

// lock an account
walletApi.lock();
```

4.3.2 Update Name

Unlock .

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// create an name
Name name = randomName();
walletApi.with(client).transaction().createName(name);

// sleep
Thread.sleep(2000L);

// update an name
AccountAddress nextOwner = AccountAddress
```

(continues on next page)

( )

```
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
TxHash txHash = walletApi.with(client).transaction().updateName(name, nextOwner);
System.out.println("Update name tx hash: " + txHash);

// lock an account
walletApi.lock();
```

4.3.3 Stake

Unlock staking .

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// stake
TxHash txHash = walletApi.with(client).transaction().stake(Aer.of("10000", Unit.
↪AERGO));
System.out.println("Stake tx hash: " + txHash);

// lock an account
walletApi.lock();
```

4.3.4 Unstake

Unlock unstaking .

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// unstake
TxHash txHash = walletApi.with(client).transaction().unstake(Aer.of("10000", Unit.
↪AERGO));
System.out.println("Unstake tx hash: " + txHash);

// lock an account
walletApi.lock();
```

4.3.5 Vote

Unlock .

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// vote to "voteBP"
List<String> candidates = asList(
↪"16Uiu2HAkwWbv8nKx7S6S5NMvUpTLNeXMVCPr3NTnrx6rBPYYiQ4K");
TxHash txHash = walletApi.with(client).transaction().vote("voteBp", candidates);
System.out.println("Vote tx hash: " + txHash);

// lock an account
walletApi.lock();
```

4.3.6 Send

Unlock aergo .

Payload .

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// send
AccountAddress accountAddress = AccountAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
TxHash txHash = walletApi.with(client).transaction()
    .send(accountAddress, Aer.AERGO_ONE, Fee.INFINITY);
System.out.println("Send tx hash: " + txHash);

// lock an account
walletApi.lock();
```

Payload .

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// send
AccountAddress accountAddress = AccountAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
BytesValue payload = BytesValue.of("test".getBytes());
TxHash txHash = walletApi.with(client).transaction()
    .send(accountAddress, Aer.AERGO_ONE, Fee.INFINITY, payload);
System.out.println("Send tx hash: " + txHash);

// lock an account
walletApi.lock();
```

Payload .

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// create an name
Name name = randomName();
walletApi.with(client).transaction().createName(name);

// sleep
Thread.sleep(2000L);

// send
TxHash txHash = walletApi.with(client).transaction()
    .send(name, Aer.AERGO_ONE, Fee.INFINITY);
System.out.println("Send tx hash: " + txHash);

// lock an account
walletApi.lock();
```

Payload .

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// create an name
Name name = randomName();
walletApi.with(client).transaction().createName(name);

// sleep
Thread.sleep(2000L);

// send
BytesValue payload = BytesValue.of("test".getBytes());
TxHash txHash = walletApi.with(client).transaction()
    .send(name, Aer.AERGO_ONE, Fee.INFINITY, payload);
System.out.println("Send tx hash: " + txHash);

// lock an account
walletApi.lock();
```

4.3.7 Commit

Unlock commit .

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// create a raw transaction
AccountAddress current = walletApi.getPrincipal();
ChainIdHash chainIdHash = walletApi.with(client).query().getChainIdHash();
AccountState currentState = walletApi.with(client).query().getAccountState(current);
RawTransaction rawTransaction = RawTransaction.newBuilder()
    .chainIdHash(chainIdHash)
    .from(current)
    .to(current)
    .amount(Aer.AERGO_ONE)
    .nonce(currentState.getNonce() + 1L)
    .build();

// commit
TxHash txHash = walletApi.with(client).transaction().commit(rawTransaction);
System.out.println("Commit tx hash: " + txHash);

// lock an account
walletApi.lock();
```

transaction commit .

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// create a signed transaction
AccountAddress current = walletApi.getPrincipal();
ChainIdHash chainIdHash = walletApi.with(client).query().getChainIdHash();
AccountState currentState = walletApi.with(client).query().getAccountState(current);
RawTransaction rawTransaction = RawTransaction.newBuilder()
    .chainIdHash(chainIdHash)
```

(continues on next page)

( )

```
    .from(current)
    .to(current)
    .amount(Aer.AERGO_ONE)
    .nonce(currentState.getNonce() + 1L)
    .build();
Transaction signed = walletApi.sign(rawTransaction);

// commit
TxHash txHash = walletApi.with(client).transaction().commit(signed);
System.out.println("Commit tx hash: " + txHash);

// lock an account
walletApi.lock();
```

4.3.8 Deploy

Unlock contract .contract definition *ContractDefinition*.

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// make a contract definition
String encodedContract = contractPayload;
ContractDefinition contractDefinition = ContractDefinition.newBuilder()
    .encodedContract(encodedContract)
    .build();

// deploy contract
TxHash txHash = walletApi.with(client).transaction().deploy(contractDefinition, Fee.
↪INFINITY);
System.out.println("Deploy tx hash: " + txHash);

// sleep
Thread.sleep(2000L);

// get ContractTxReceipt
ContractTxReceipt contractTxReceipt = walletApi.with(client).query()
    .getContractTxReceipt(txHash);
System.out.println("Deployed contract tx receipt: " + contractTxReceipt);

// get contract interface
ContractAddress contractAddress = contractTxReceipt.getContractAddress();
ContractInterface contractInterface = walletApi.with(client).query()
    .getContractInterface(contractAddress);
System.out.println("Deployed contract interface: " + contractInterface);

// lock an account
walletApi.lock();
```

4.3.9 Re-Deploy

Unlock contract .contract definition *ContractDefinition*.

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// made by aergoluac --compiledContract {some_contract}.lua
String encodedContract = contractPayload;

// make a contract definition
ContractDefinition newDefinition = ContractDefinition.newBuilder()
    .encodedContract(encodedContract)
    .build();

// redeploy
ContractAddress contractAddress = contractAddressKeep;
TxHash txHash = walletApi.with(client).transaction()
    .redeploy(contractAddress, newDefinition, Fee.INFINITY);
System.out.println("Redeploy tx hash: " + txHash);

// lock an account
walletApi.lock();
```

4.3.10 Execute

Unlock contract .contract invocation *ContractInvocation*.

```
// unlock specific account with authentication
walletApi.unlock(authentication);

// make a contract invocation
ContractInterface contractInterface = contractInterfaceKeep;
ContractInvocation contractInvocation = contractInterface.newInvocationBuilder()
    .function("set")
    .args("key", 333, "test2")
    .build();

// execute
TxHash txHash = walletApi.with(client).transaction()
    .execute(contractInvocation, Fee.INFINITY);
System.out.println("Execute tx hash: " + txHash);

// lock an account
walletApi.lock();
```

4.4 Query Api

QueryApi high-level api . unlock .

4.4.1 Get Account State

.

```
// get account state
AccountAddress accountAddress = AccountAddress
```

(continues on next page)

( )

```
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
AccountState state = walletApi.with(client).query()
    .getAccountState(accountAddress);
System.out.println("Account state: " + state);
```

4.4.2 Get Name Owner

.

.

```
// get name owner
Name name = Name.of("namenamename");
AccountAddress nameOwner = walletApi.with(client).query().getNameOwner(name);
System.out.println("Name owner: " + nameOwner);
```

.

```
// get name owner at block 10
Name name = Name.of("namenamename");
AccountAddress nameOwner = walletApi.with(client).query().getNameOwner(name, 10);
System.out.println("Name owner: " + nameOwner);
```

4.4.3 Get Stake Info

staking .

```
// get stake info
AccountAddress accountAddress = AccountAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
StakeInfo stakeInfo = walletApi.with(client).query().getStakeInfo(accountAddress);
System.out.println("Stake info: " + stakeInfo);
```

4.4.4 List Elected Bps

Block producers .

```
// list elected bps
List<ElectedCandidate> candidates = walletApi.with(client).query().listElectedBps(23);
System.out.println("Elected bps: " + candidates);
```

4.4.5 List Elected

Vote id .

```
// list elected for "voteBP"
List<ElectedCandidate> candidates = walletApi.with(client).query()
    .listElected("voteBP", 23);
System.out.println("Elected candidates: " + candidates);
```

4.4.6 Get Vote Info

.

```
// get vote info
AccountAddress accountAddress = AccountAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
AccountTotalVote accountTotalVote = walletApi.with(client).query().
↪getVotesOf(accountAddress);
System.out.println("Account total vote: " + accountTotalVote);
```

4.4.7 Get Best Block Hash

node .

```
// get best block hash
BlockHash blockHash = walletApi.with(client).query().getBestBlockHash();
System.out.println("Best block hash: " + blockHash);
```

4.4.8 Get Best Block Height

node .

```
// get best block hash
long blockHeight = walletApi.with(client).query().getBestBlockHeight();
System.out.println("Best block height: " + blockHeight);
```

4.4.9 Get Chain Id Hash

node chain id hash .

```
// get chain id hash
ChainIdHash chainIdHash = walletApi.with(client).query().getChainIdHash();
System.out.println("Chain id hash: " + chainIdHash);
```

4.4.10 Get Blockchain Status

node blockchain .

```
// get blockchain status
BlockchainStatus blockchainStatus = walletApi.with(client).query().
↪getBlockchainStatus();
System.out.println("Blockchain status: " + blockchainStatus);
```

4.4.11 Get Chain Info

node chain .

```
// get chain info
ChainInfo chainInfo = walletApi.with(client).query().getChainInfo();
System.out.println("ChainInfo: " + chainInfo);
```

4.4.12 Get Chain Stats

node chain .

```
// get chain stats
ChainStats chainStats = walletApi.with(client).query().getChainStats();
System.out.println("ChainStats: " + chainStats);
```

4.4.13 List Peers

node peer .

hidden peer .

```
// list peers
List<Peer> peers = walletApi.with(client).query().listPeers();
System.out.println("Peers: " + peers);
```

hidden peer .

```
// list peers
List<Peer> peers = walletApi.with(client).query().listPeers(true, true);
System.out.println("Peers: " + peers);
```

4.4.14 List Peer Metrics

node peer .

```
// list peer metrics
List<PeerMetric> peerMetrics = walletApi.with(client).query().listPeerMetrics();
System.out.println("Peer metrics: " + peerMetrics);
```

4.4.15 Get Server Info

node . Category .

```
// get server info
List<String> categories = emptyList();
ServerInfo serverInfo = walletApi.with(client).query().getServerInfo(categories);
System.out.println("Server info: " + serverInfo);
```

4.4.16 Get Node Status

node .

```
// get node status
NodeStatus nodeStatus = walletApi.with(client).query().getNodeStatus();
System.out.println("Node status: " + nodeStatus);
```

4.4.17 Get Block Metadata

Block . null .

.

```
// get block metadata
BlockHash blockHash = BlockHash.of("DN9TvryaThbJneSpzaXp5ZsS4gE3UMzKfaXC4x8L5qR1");
BlockMetadata blockMetadata = walletApi.with(client).query().
↪getBlockMetadata(blockHash);
System.out.println("Block metadata by hash: " + blockMetadata);
```

.

```
// get block metadata
long height = 27_066_653L;
BlockMetadata blockMetadata = walletApi.with(client).query().getBlockMetadata(height);
System.out.println("Block metadata by height: " + blockMetadata);
```

4.4.18 List Block Metadata

Block . 1000.

.

```
// block metadatas by from hash to previous 100 block
BlockHash blockHash = BlockHash.of("DN9TvryaThbJneSpzaXp5ZsS4gE3UMzKfaXC4x8L5qR1");
List<BlockMetadata> blockMetadatas = walletApi.with(client).query()
    .listBlockMetadatas(blockHash, 100);
System.out.println("Block metadatas by hash: " + blockMetadatas);
```

.

```
// block metadatas by from height to previous 100 block
long height = 27_066_653L;
List<BlockMetadata> blockMetadatas = walletApi.with(client).query()
    .listBlockMetadatas(height, 100);
System.out.println("Block metadatas by height: " + blockMetadatas);
```

4.4.19 Get Block

Block . null .

.

```
// get block by hash
BlockHash blockHash = BlockHash.of("DN9TvryaThbJneSpzaXp5ZsS4gE3UMzKfaXC4x8L5qR1");
Block block = walletApi.with(client).query().getBlock(blockHash);
System.out.println("Block by hash: " + block);
```

.

```
// get block by height
long height = 27_066_653L;
Block block = walletApi.with(client).query().getBlock(height);
System.out.println("Block by hash: " + block);
```

4.4.20 Block Metadata Subscription

Block .

```
// make a subscription
Subscription<BlockMetadata> metadataSubscription = walletApi.with(client).query()
    .subscribeBlockMetadata(new StreamObserver<BlockMetadata>() {
      @Override
      public void onNext(BlockMetadata value) {
        System.out.println("Next block metadata: " + value);
      }

      @Override
      public void onError(Throwable t) {

      }

      @Override
      public void onCompleted() {
      }
    });

// wait for a while
Thread.sleep(2000L);

// unsubscribe it
metadataSubscription.unsubscribe();
```

4.4.21 Block Subscription

Block .

```
// make a subscription
Subscription<Block> subscription = walletApi.with(client).query()
    .subscribeBlock(new StreamObserver<Block>() {
      @Override
      public void onNext(Block value) {
        System.out.println("Next block: " + value);
      }

      @Override
      public void onError(Throwable t) {
      }

      @Override
      public void onCompleted() {
      }
    });
```

(continues on next page)

( )

```
// wait for a while
Thread.sleep(2000L);

// unsubscribe it
subscription.unsubscribe();
```

4.4.22 Get Transaction

Transaction . null .

```
// get transaction
TxHash txHash = TxHash.of("39vLyMqsg1mTT9mF5NbADgNB2YUiRVsT6SUkDujBZme8");
Transaction transaction = walletApi.with(client).query().getTransaction(txHash);
System.out.println("Transaction: " + transaction);
```

4.4.23 Get Transaction Receipt

Transaction . null .

```
// get tx receipt
TxHash txHash = TxHash.of("39vLyMqsg1mTT9mF5NbADgNB2YUiRVsT6SUkDujBZme8");
TxReceipt txReceipt = walletApi.with(client).query().getTxReceipt(txHash);
System.out.println("Transaction receipt: " + txReceipt);
```

4.4.24 Get Contract Tx Receipt

contract tx . null .

```
// get contract tx receipt
TxHash txHash = TxHash.of("EGXNDgjY2vQ6uuP3UF3dNXud54dF4FNVY181kaeQ26H9");
ContractTxReceipt contractTxReceipt = walletApi.with(client).query()
    .getContractTxReceipt(txHash);
System.out.println("Contract tx receipt: " + contractTxReceipt);
```

4.4.25 Get Contract Interface

contract interface . null .

```
// get contract interface
ContractAddress contractAddress = ContractAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
ContractInterface contractInterface = walletApi.with(client).query()
    .getContractInterface(contractAddress);
System.out.println("ContractInterface: " + contractInterface);
```

4.4.26 Query Contract

contract . Contract invocation *ContractInvocation*.

```
// make a contract invocation
ContractInterface contractInterface = contractInterfaceKeep;
ContractInvocation query = contractInterface.newInvocationBuilder()
    .function("get")
    .args("key")
    .build();

// query contract
ContractResult queryResult = client.getContractOperation().query(query);
Data data = queryResult.bind(Data.class);
System.out.println("Raw contract result: " + queryResult);
System.out.println("Binded data: " + data);
```

4.4.27 List Event

block event . Event filter *EventFilter*.

```
// list events with a filter
ContractAddress contractAddress = contractAddressKeep;
EventFilter eventFilter = EventFilter.newBuilder(contractAddress)
    .eventName("set")
    .args("key")
    .recentBlockCount(1000)
    .build();
List<Event> events = client.getContractOperation().listEvents(eventFilter);
System.out.println("Events: " + events);
```

4.4.28 Event Subscription

event . Event filter *EventFilter*.

```
// subscribe event
ContractAddress contractAddress = ContractAddress
    .of("AmNrsAqkXhQfE6sGxTutQkf9ekaYowaJFLekEm8qvDr1RB1AnsiM");
EventFilter eventFilter = EventFilter.newBuilder(contractAddress)
    .recentBlockCount(1000)
    .build();
Subscription<Event> subscription = client.getContractOperation()
    .subscribeEvent(eventFilter, new StreamObserver<Event>() {
      @Override
      public void onNext(Event value) {
        System.out.println("Next event: " + value);
      }

      @Override
      public void onError(Throwable t) {
      }

      @Override
      public void onCompleted() {
      }
    });

Thread.sleep(2200L);
```

(continues on next page)

( )

```
// unsubscribe event
subscription.unsubscribe();
```

CHAPTER 5

Contract

5.1 Contract Api

ContractApi java interface contract call . ContractApi nonce . nonce nonce nonce .

5.1.1 Prepare

ContractApi , smart contract . smart contract interface .

smart contract . Programming Guide.

```
function constructor(key, arg1, arg2)
  if key ~= nil then
    system.setItem(key, {intVal=arg1, stringVal=arg2})
  end
end

function set(key, arg1, arg2)
  contract.event("set", key, arg1, arg2)
  system.setItem(key, {intVal=arg1, stringVal=arg2})
end

function get(key)
  return system.getItem(key)
end

function check_delegation()
  return true
end

abi.register_view(get)
abi.register(set)
abi.fee_delegation(set)
abi.payable(set)
```

smart contract .

```
// make a contract definition
String encodedContract = contractPayload;
ContractDefinition contractDefinition = ContractDefinition.newBuilder()
    .encodedContract(encodedContract)
    .build();

// deploy contract
walletApi.unlock(authentication);
TxHash txHash = walletApi.with(client).transaction()
    .deploy(contractDefinition, Fee.INFINITY);
walletApi.lock();

// sleep
Thread.sleep(2000L);

// get ContractTxReceipt
ContractTxReceipt contractTxReceipt = walletApi.with(client).query()
    .getContractTxReceipt(txHash);

// get contract address
ContractAddress contractAddress = contractTxReceipt.getContractAddress();
System.out.println("Deployed contract address: " + contractPayload);
```

interface . method smart contract .

```
// interface for smart contract
interface CustomInterface1 {

  /*
    Matches with

      function set(key, arg1, arg2)
        ...
      end

      ...

      abi.register(set)

    And it also uses provided fee when making transaction.
   */
  TxHash set(String key, int arg1, String args2, Fee fee);

  /*
    Matches with

      function set(key, arg1, arg2)
        ...
      end

      ...

      abi.register(set)

    And it also uses Fee.INFINITY when making transaction.
   */
```

(continues on next page)

( )

```
  TxHash set(String key, int arg1, String args2);

  /*
    Matches with

      function get(key)
        ...
        -- returns lua table which can be binded with Data class
        return someVal
      end

      ...

      abi.register_view(get)
   */
  Data get(String key);

}

// java bean
class Data {

  protected int intVal;

  protected String stringVal;

  public int getIntVal() {
    return intVal;
  }

  public void setIntVal(int intVal) {
    this.intVal = intVal;
  }

  public String getStringVal() {
    return stringVal;
  }

  public void setStringVal(String stringVal) {
    this.stringVal = stringVal;
  }

  @Override
  public String toString() {
    return "Data{" +
        "intVal=" + intVal +
        ", stringVal=" + stringVal +
        '}';
  }
}
```

5.1.2 Make

smart contract interface ContractApi .

nonce .

```
// create a contract api
ContractAddress contractAddress = deployedContractAddress;
ContractApi<CustomInterface1> contractApi = new ContractApiFactory()
    .create(contractAddress, CustomInterface1.class);
System.out.println("ContractApi: " + contractApi);
```

nonce .

```
// create a contract api with retry count 5 and interval 1000ms
ContractAddress contractAddress = deployedContractAddress;
TryCountAndInterval tryCountAndInterval = TryCountAndInterval.of(5, Time.of(1000L));
ContractApi<CustomInterface1> contractApi = new ContractApiFactory()
    .create(contractAddress, CustomInterface1.class, tryCountAndInterval);
System.out.println("ContractApi: " + contractApi);
```

5.1.3 Execute

AergoKey .

```
// prepare an signer
AergoKey signer = richKey;

// create a contract api
ContractAddress contractAddress = deployedContractAddress;
ContractApi<CustomInterface1> contractApi = new ContractApiFactory()
    .create(contractAddress, CustomInterface1.class);

// execute contract with a contract api
TxHash executeTxHash = contractApi.with(client).execution(signer)
    .set("key", 123, "test", Fee.INFINITY);
System.out.println("Execute tx hash: " + executeTxHash);
```

WalletApi .

```
// create a contract api
ContractAddress contractAddress = deployedContractAddress;
ContractApi<CustomInterface1> contractApi = new ContractApiFactory()
    .create(contractAddress, CustomInterface1.class);

// execute contract with a contract api
walletApi.unlock(authentication);
TxHash executeTxHash = contractApi.with(client).execution(walletApi)
    .set("key", 123, "test", Fee.INFINITY);
walletApi.lock();
System.out.println("Execute tx hash: " + executeTxHash);
```

5.1.4 Query

Binging model .

```
// create a contract api
ContractAddress contractAddress = deployedContractAddress;
ContractApi<CustomInterface1> contractApi = new ContractApiFactory()
    .create(contractAddress, CustomInterface1.class);
```

(continues on next page)

( )

```
// query contract with a contract api
Data data = contractApi.with(client).query().get("key");
System.out.println("Queried data: " + data);
```

Binging model .

```
// create a contract api
ContractAddress contractAddress = deployedContractAddress;
ContractApi<CustomInterface2> contractApi = new ContractApiFactory()
    .create(contractAddress, CustomInterface2.class);

// query contract with a contract api
ContractResult contractResult = contractApi.with(client).query().get("key");
System.out.println("Queried data: " + contractResult);
```

CHAPTER 6

Examples

https://github.com/aergoio/heraj-example