HCX Send/Receive Tutorial

This tutorial is divided into sections shown below:

1. HCNet Services:- We will provide three services to you
    a. createAccount: For HCX address generation (Please review Appendix d (i))
    b. fundTransfer: For HCX send (Please review Appendix d (ii))
    c. getBalance: For balance check (Please review Appendix d (iii))

2. Pool Account Management:- You need to create an HCX Pool Accountfor your exchange. You can do this by using the createAccount service mentioned above (and described below). Only you would hold the private key of this Pool Account. You can fund this Pool Account by either buying HCX from another exchange or requesting us to send HCX to your Pool Account. You can buy HCX by registering an account on PayBito exchange hosted at https://trade.paybito.com. If you have bought HCX at PayBito or another exchange, you will need to transfer the HCX from that exchange's account to your Pool Account address. You can create any number of accounts using the createAccount service and transfer funds between them using fundTransfer service (you only need one Pool Account for integrating HCX to your exchange).

3. *Accounting Service:-*
    - *Create a table which holds customer records, you may use below columns in your table: customer_id, hcx_pubkey, hcx_privkey, hcx_balance as an example.*
    - *After the table is created, insert pool account details (customer_id, hcx_pubkey, hcx_privkey, hcx_balance) as mentioned above in your DB.*

Note: This section shows an example exchange integration. Actual code for your exchange application could differ.

Now you need to integrate HCNet services for your Exchange to Send/Receive HCX. Please use the steps identified below.

A. How to generate HCX address for your exchange customer using createAccount service

You need to first check whether your customer has HCX address or not. If not then call createAccountService as describe in Appendix b (i).

Note:- To generate HCX address for the first time, the network requires you to have 20 HCX. We will supply this automatically to you from our HCX account.

An important point to note here is that, you should keep the HCX balance to 0 in your DB table and update the table when customer receives HCX.

B. How HCX is sent to same wallet and other wallet

First check balance from your table(column - HCX_BALANCE) regarding customer_id which will send HCX to other customer,if balance is sufficient then customer can send HCX to other customers.

Note: In this case HCX will be deducted from pool account on behalf of sender. Please check Appendix b (ii) to get help.

C. Display Pool account balance to your admin

Call getBalance api to display balance of pool account. Go to Appendix c.

Note: This section shows an example exchange integration. Actual code for your exchange application could differ.

D. Database Design for HCX Notification:-

1. *Create a table i.e HCX_RECEIVED to store all received HCX details.*
2. *Create a sequence i.e HCX_RECEIVED_SEQ to get auto increment id.*
3. *Create a trigger i.e HCX_RECEIVED_TRG to get sequence number from HCX_RECEIVED_SEQ.*
4. *Create one another table i.e HCX_BALANCE to maintain HCX balance of all customers.*
5. *Create a trigger i.e UPDATE_CUSTOMER_HCX_BALANCE on HCX_RECEIVED table to update HCX_BALANCE table.*

   Note:- When data is inserted into HCX_RECEIVED table, UPDATE_CUSTOMER_HCX_BALANCE trigger will fire.

Note: This section shows an example exchange integration. Actual code for your exchange application could differ.

E. Receiving HCX

In HCNet, transactions happen within 2-5 seconds. So you do not need to wait too long for receiving HCX. To know about payment notification you can use HCXNotification as described below. In HCXNotification you will be notified immediately when transaction happens in HC Net. You need to validate if there is a corresponding customer in your exchange for that transaction. If the customer exists, then update the customer balance and transfer the payment to your pool account after deducting the network transaction fee i.e 0.0001 HCX. You need to perform below steps to accomplish this.

1. Create table like HCX_BALANCE for maintaining HCX amount from node:

```
CREATE TABLE "HCX_BALANCE"
   ("CUSTOMER_ID" NUMBER(10 BYTE),
    "PUBKEY" VARCHAR2(100 BYTE),
    "PRIVATEKEY" VARCHAR2(100 BYTE),
    "HCX_AMOUNT" NUMBER(20,10) DEFAULT 0,
     PRIMARY KEY ("CUSTOMER_ID ")
  );
```

2. Create table like HCX_RECEIVED for receiving HCX amount from node:

```
CREATE TABLE "HCX_RECEIVED"
   ("ID" NUMBER (10 BYTE) NOT NULL ENABLE,
    "CUSTOMER_ID" NUMBER(10 BYTE),
    "PUBKEY" VARCHAR2(100 BYTE),
    "PRIVATEKEY" VARCHAR2(100 BYTE),
    "HCX_AMOUNT" NUMBER(20,10) DEFAULT 0,
    "TXNID" VARCHAR2(100 BYTE),
    "CREATED_ON" TIMESTAMP (6),
     PRIMARY KEY ("ID")
  );
```

3. Create table like CUSTOMERHCXKEYS for HCX sending purpose:

```
CREATE TABLE "CUSTOMERHCXKEYS"

             ("ID" NUMBER(10,0) NOT NULL ENABLE,

             "CUSTOMERID" VARCHAR2(50 BYTE) NOT NULL ENABLE,
             "FROMADD" VARCHAR2(100 BYTE),
             "TOADD" VARCHAR2(100 BYTE),
             "HCXAMOUNT" NUMBER(30,10),
             "TXNID" VARCHAR2(100 BYTE),
             "STATUS" VARCHAR2(100 BYTE),
             "ACTION" VARCHAR2(50 BYTE),
             "CURRENCY" VARCHAR2(10 BYTE),
             "LEDGER" NUMBER(20,0)
              );
```

4. Create a sequence PAYMENT_HISTORY for auto HCX sending purpose:

```
CREATE TABLE "PAYMENT_HISTORY"
      ("TRANSACTIONID" NUMBER(10,0) NOT NULL ENABLE,
       "CUSTOMER_ID" NUMBER(10,0),
       "ORDERID" VARCHAR2(100 BYTE),
       "MININGFEES" NUMBER(30,20) DEFAULT 0,
       "TXNCHARGE" NUMBER(30,20) DEFAULT 0,
       "SGST" NUMBER(30,20) DEFAULT 0,
       "CGST" NUMBER(30,20) DEFAULT 0,
       "TRANSACTION_TIMESTAMP" TIMESTAMP (6),
       "DESCRIPTION" VARCHAR2(255 BYTE),
       "ACTION" VARCHAR2(10 BYTE),
       "STATUS" NUMBER(2,0) DEFAULT 0,
       "CURRENCY" VARCHAR2(20 BYTE),
```

```
"NETWORKFEES" NUMBER(30,20) DEFAULT 0,
"TRADE" NUMBER(2,0) DEFAULT 0,
"BASECURRENCY" VARCHAR2(10 BYTE) DEFAULT NULL,
"TXNCHARGE_CRYPTO" NUMBER(30,20) DEFAULT 0,
"HCX_TXNID" VARCHAR2(100 BYTE),
"DEBIT_HCX_AMOUNT" NUMBER(30,8) DEFAULT 0,
"CREDIT_HCX_AMOUNT" NUMBER(30,8) DEFAULT 0,
"CLOSING_HCX_BAL" NUMBER(30,8) DEFAULT 0
);
```

5. Create a sequence HCX_RECEIVED_SEQ for auto generate id in HCX_RECEIVED table

```
CREATE SEQUENCE "HCX_RECEIVED_SEQ"
MINVALUE 1
MAXVALUE 9999999999
INCREMENT BY 1
START WITH 1 NOCACHE NOORDER NOCYCLE;
```

6. Create a trigger on HCX_RECEIVED table auto generate id in HCX_RECEIVED table:

```
CREATE OR REPLACE TRIGGER "HCX_RECEIVED_TRG"
BEFORE INSERT ON HCX_RECEIVED
FOR EACH ROW
BEGIN
  <<COLUMN_SEQUENCES>>
  BEGIN
    IF INSERTING AND :NEW.ID IS NULL THEN
      SELECT HCX_RECEIVED_SEQ.NEXTVAL INTO :NEW.ID FROM SYS.DUAL;
    END IF;
  END COLUMN_SEQUENCES;
END;
/
ALTER TRIGGER "HCX_RECEIVED_TRG" ENABLE;
```

7. Create another trigger on HCX_RECEIVED table to update customer's HCX balance:

```
CREATE OR REPLACE TRIGGER "UPDATE_CUSTOMER_HCX_BALANCE"
AFTER INSERT ON HCX_RECEIVED FOR EACH ROW

DECLARE
VAR_CUSTOMER_ID          NUMBER(10);
VAR_HCX_AMOUNT           NUMBER(20,10);
VAR_PUBKEY               VARCAHR2(100);
VAR_PRIVKEY              VARCAHR2(100);


VAR_EXST_CUSTID          NUMBER :=0 ;
VAR_EXST_CUSTID_HCX      NUMBER :=0 ;

BEGIN
    VAR_CUSTOMER_ID      := :NEW.CUSTOMER_ID;
    VAR_HCX_AMOUNT       := :NEW.HCX_AMOUNT;
VAR_PUBKEY      := :NEW.PUBKEY;
    VAR_PRIVKEY       := :NEW.PRIVATEKEY;

    IF VAR_HCX_AMOUNT > 0 THEN

       SELECT CASE WHEN (EXISTS ( SELECT * FROM HCX_BALANCE
       WHERE CUSTOMER_ID = VAR_CUSTOMER_ID )) THEN 1 ELSE 0 END
       INTO VAR_EXST_CUSTID FROM DUAL;
```

```
            IF VAR_EXST_CUSTID = 0 THEN

                    UPDATE HCX_BALANCE
                    SET HCX_AMOUNT = HCX_AMOUNT + VAR_HCX_AMOUNT
                    WHERE CUSTOMER_ID = VAR_CUSTOMER_ID;

            ELSE

                    INSERT INTO
HCX_BALANCE(CUSTOMER_ID,PUBKEY,PRIVATEKEY,HCX_AMOUNT)

                    VALUES(VAR_CUSTOMER_ID,VAR_PUBKEY,VAR_PRIVKEY,VAR_HCX_AMOUNT);

            END IF;
        END IF;
    END;
    /
    ALTER TRIGGER "UPDATE_CUSTOMER_HCX_BALANCE" ENABLE;
```

8. Go to Appendix a to check HCXNotifier code.

F. Appendix:

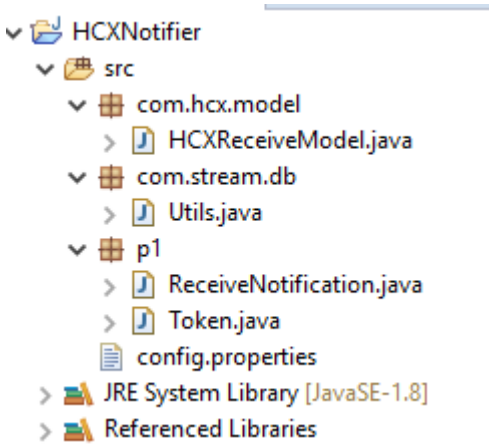
All code sample of Accounting services:-

a) HCXNotifier

Project Structure

For Notifier you should keep cursor in your local file. Because through cursor, you would control your notifier. Otherwise the notifier will stream from current position.

Token.java(To save network payment cursor in localdrive) create a file called pagingtoken.txt inside the project folder.



config.properties

dburl= Define database url.
dbuser= Define database username.
dbpassword= Define database password.
hcxkey= Define pool account public key.
network_passphrase= Define network password .
aurora_url= https://bitpaymentz.com
user= Define pool account CUSTOMER_ID.
ledgertable=HCX_BALANCE (example table name)
hcxtable=HCX_RECEIVED (example table name)

HCXReceiveModel.java

```
package com.hcx.model;
publicclass HCXReceiveModel {
private String customer_id;
private String hcx_privkey;
private String hcx_pubkey;
public String getCustomer_id() {
returncustomer_id;
```

```
}
publicvoid setCustomer_id(String customer_id) {
this.customer_id = customer_id;
}
public String getHcx_privkey() {
returnhcx_privkey;
}
publicvoid setHcx_privkey(String hcx_privkey) {
this.hcx_privkey = hcx_privkey;
}
public String getHcx_pubkey() {
returnhcx_pubkey;
}
publicvoid setHcx_pubkey(String hcx_pubkey) {
this.hcx_pubkey = hcx_pubkey;
}
}
```

Utils.java

```
package com.stream.db;
import java.io.IOException;
import java.io.InputStream;
import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.Properties;
import javax.sql.PooledConnection;
import com.hcx.model.HCXReceiveModel;
import oracle.jdbc.pool.OracleConnectionPoolDataSource;

public class Utils {
Properties prop = new Properties();
public static OracleConnectionPoolDataSource ocpds=null;
public  static Properties prop1 = new Properties();
public static InputStream input = null;
public  static PooledConnection pc=null;
public static Connection con =null;
public static PreparedStatement pstmt=null;
public static ResultSet rs =null;
public static ArrayList<HCXReceiveModel>accounts=null;
public  static String pooluser=null;
public static String ledgertable=null;
public static String hcxtable=null;
```

```
public  static boolean flag=false;

static{
try {
String filename = "config.properties";
input =Utils.class.getClassLoader().getResourceAsStream(filename);
prop1.load(input);
String dburl=prop1.getProperty("dburl");
String dbuser=prop1.getProperty("dbuser");
String dbpassword=prop1.getProperty("dbpassword");
pooluser=prop1.getProperty("user");
ledgertable=prop1.getProperty("ledgertable");
hcxtable=prop1.getProperty("hcxtable");
ocpds=new OracleConnectionPoolDataSource();
ocpds.setURL(dburl);
ocpds.setUser(dbuser);
ocpds.setPassword(dbpassword);
} catch (SQLException e) {
e.printStackTrace();
} catch (IOException e) {
e.printStackTrace();
}
}

public static ArrayList<HCXReceiveModel> getAccountHCXPool(){
accounts=new ArrayList<>();
try {
pc=ocpds.getPooledConnection();
con= pc.getConnection();
String sql ="select customer_id,hcx_privkey,hcx_pubkey from "+ledgertable+" where not
customer_id ='"+pooluser+"' and hcx_privkey is not null";
pstmt = con.prepareStatement(sql);
rs = pstmt.executeQuery();
while(rs.next()){
HCXReceiveModel  hcxModel=new  HCXReceiveModel();
hcxModel.setCustomer_id(rs.getString("customer_id"));
hcxModel.setHcx_privkey(rs.getString("hcx_privkey"));
hcxModel.setHcx_pubkey(rs.getString("hcx_pubkey"));
accounts.add(hcxModel);
}
}
catch(Exception e){
e.printStackTrace();
}
finally{
if(con!=null)
{
```

```
try {
pstmt.close();
con.close();
pc.close();
} catch (SQLException e) {
e.printStackTrace();
}
}
}
return accounts;
}

public static boolean insertuser(String custid,String pubkey,String privkey,BigDecimal
amount,String txnid,Connection con){
try {
java.util.Date utilDate = new java.util.Date();
Long currenttime=utilDate.getTime();
pstmt = con.prepareStatement("insert into
"+hcxtable+"(CUSTOMER_ID,PUBKEY,PRIVATEKEY,HCX_AMOUNT,TXNID,CREATED_ON)
values(?,?,?,?,?,?)");
pstmt.setString(1,custid);
pstmt.setString(2,pubkey);
pstmt.setString(3,privkey);
pstmt.setBigDecimal(4,amount);
pstmt.setString(5,txnid);
pstmt.setTimestamp(6,(new java.sql.Timestamp(currenttime)));
int status=pstmt.executeUpdate();
if(status>0){
flag=true;
}
Else
{
flag=false;
}

}
catch(Exception e){
e.printStackTrace();
}
finally{
if(con!=null){
try {
pstmt.close();
con.close();
pc.close();
} catch (SQLException e) {
e.printStackTrace();
```

```
}
}
}
return flag;
}
public static Connection getConnection(){
try {
pc=ocpds.getPooledConnection();
con= pc.getConnection();
} catch (SQLException e) {
e.printStackTrace();
}
return con;
}
}
```

Token.java

```
package p1;

import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.URL;
import java.util.Properties;
import java.util.StringTokenizer;
import org.json.JSONObject;

public class Token {
static Properties prop = new Properties();
static String passphrase=null;
static InputStream input = null;
static String ptoken=null;
static String url=null;
static FileReader fileReader=null;
static  BufferedReader bufferedReader=null;
static{
try{
String filename = "config.properties";
input = ReceiveNotification.class.getClassLoader().getResourceAsStream(filename);

prop.load(input);
```

```
url = prop.getProperty("aurora_url");
}catch(Exception e){
e.printStackTrace();
}
}
static void  getpagingtoken(){
try{
fileReader=new FileReader("pagingtoken.txt");
bufferedReader=new BufferedReader(fileReader);
ptoken=bufferedReader.readLine();
bufferedReader.close();
fileReader.close();
if(ptoken.length()==0)
{
String myurl = url+"/payments?order=desc";
System.out.println(myurl);
URL url = new URL(myurl);
HttpURLConnection connection = (HttpURLConnection) url.openConnection();
connection.connect();
InputStream inputStream = connection.getInputStream();
InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
String line = "";
StringBuffer buffer = new StringBuffer();
while ((line = bufferedReader.readLine()) != null) {
buffer.append(line);
}
System.out.println(" ");
String finalJSON = buffer.toString();
JSONObject jsonObject = new JSONObject(finalJSON);
JSONObject jsonObject1 = jsonObject.getJSONObject("_links");
JSONObject jsonObject2 = jsonObject1.getJSONObject("prev");
String next = jsonObject2.getString("href");
System.out.println();
System.out.println(next);
StringTokenizer stringTokenizer = new StringTokenizer(next, "= \u0026");
String[] token = new String[6];
while (stringTokenizer.hasMoreTokens()) {
for (int i = 0; i < token.length; i++) {
token[i] = stringTokenizer.nextToken();
}
}
String tokenid = token[1];
System.out.println("Your Operation Id = " + tokenid);
FileWriter fileWriter;
try {
fileWriter = new FileWriter("pagingtoken.txt");
```

```
BufferedWriter bufferedWriter=new BufferedWriter(fileWriter);
bufferedWriter.write(tokenid);
bufferedWriter.flush();
bufferedWriter.close();
fileWriter.close();
} catch (IOException e) {
e.printStackTrace();
}
}
else{
System.out.println("Token alraedy exists");
}
}
catch(Exception e){
e.printStackTrace();
}
}
}
```

ReceiveNotification.java

```
package p1;
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.math.BigDecimal;
import java.sql.Connection;
import java.sql.SQLException;
import java.util.List;
import java.util.Properties;
import java.util.logging.Logger;
import org.stellar.sdk.AssetTypeNative;
import org.stellar.sdk.KeyPair;
import org.stellar.sdk.Memo;
import org.stellar.sdk.Network;
import org.stellar.sdk.PaymentOperation;
import org.stellar.sdk.Server;
import org.stellar.sdk.Transaction;
import org.stellar.sdk.requests.EventListener;
import org.stellar.sdk.responses.AccountResponse;
import org.stellar.sdk.responses.SubmitTransactionResponse;
import org.stellar.sdk.responses.operations.OperationResponse;
import org.stellar.sdk.responses.operations.PaymentOperationResponse;
import com.hcx.model.HCXReceiveModel;
```

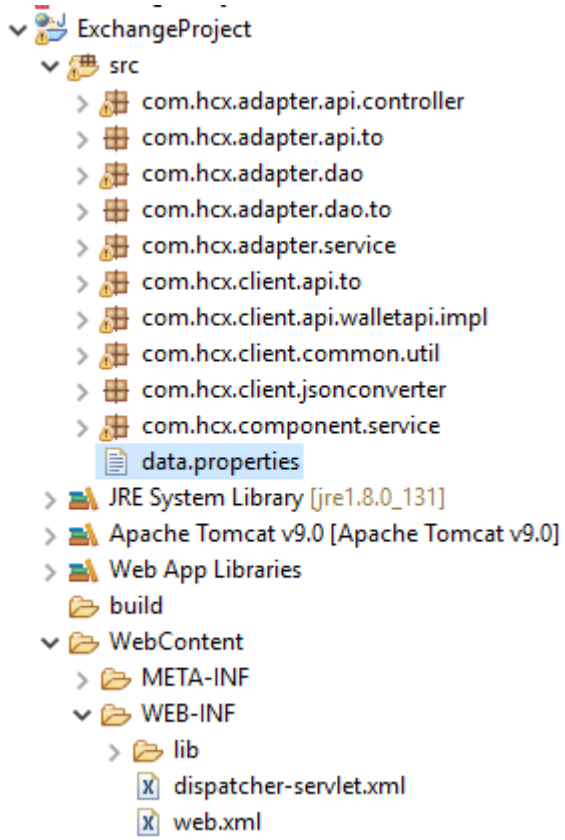
```
import com.stream.db.Utils;

public class ReceiveNotification {
static Logger logger = Logger.getLogger(ReceiveNotification.class.getName());
static Properties prop = new Properties();
static String ptoken = null;
static List<HCXReceiveModel> list = null;
static String custid = null;
static String privKey = null;
static String pubKey = null;
static String hcx_balance = null;
static BigDecimal networktxn_fee = new BigDecimal("0.00001");
static Connection con = null;
static String paybitohcx_user = null;
static FileReader fileReader = null;
static BufferedReader bufferedReader = null;
public static void main(String[] args) {
try {
InputStream input = null;
String filename = "config.properties";
input = ReceiveNotification.class.getClassLoader().getResourceAsStream(filename);
prop.load(input);
String url = prop.getProperty("aurora_url");
Server server = new Server(url);
String passphrase = prop.getProperty("network_passphrase");
paybitohcx_user = prop.getProperty("hcxkey");
Network network = new Network(passphrase);
Network.use(network);
Token.getpagingtoken();
fileReader = new FileReader("pagingtoken.txt");
bufferedReader = new BufferedReader(fileReader);
ptoken = bufferedReader.readLine();
bufferedReader.close();
fileReader.close();
HCXReceiveModel h1 = new HCXReceiveModel();
h1.setHcx_pubkey("NoAccount");
server.payments().cursor(ptoken).stream(new EventListener<OperationResponse>() {
@Override
public void onEvent(OperationResponse payment) {
System.out.println("Paging Token from file.." + ptoken);
list = Utils.getAccountHCXPool();
if (payment instanceof PaymentOperationResponse) {
System.out.println("Paging Token.." + payment.getPagingToken());
FileWriter fileWriter;
try {
fileWriter = new FileWriter("pagingtoken.txt");
BufferedWriter bufferedWriter = new BufferedWriter(fileWriter);
```

```
bufferedWriter.write(payment.getPagingToken());
bufferedWriter.flush();
bufferedWriter.close();
fileWriter.close();
fileReader = new FileReader("pagingtoken.txt");
BufferedReader bufferedReader = new BufferedReader(fileReader);
ptoken = bufferedReader.readLine();
bufferedReader.close();
fileReader.close();
} catch (IOException e) {
e.printStackTrace();
}
// System.out.println("Payment Hash"+ payment.getTransactionHash());
org.stellar.sdk.KeyPair from = ((PaymentOperationResponse) payment).getFrom();
org.stellar.sdk.KeyPair to = ((PaymentOperationResponse) payment).getTo();
System.out.println("Sender.." + from.getAccountId());
String receiver = to.getAccountId();
System.out.println("Receiver.." + receiver);
System.out.println(payment.getCreatedAt());
HCXReceiveModel hcxPaybito = list.stream().filter(h -> receiver.equals(h.getHcx_pubkey()))
.findAny().orElse(h1);
if (hcxPaybito.getHcx_pubkey().equals("NoAccount")) {
System.out.println("HCX Customer Not Found..." + hcxPaybito.getHcx_pubkey());
} else {
try {
System.out.println("HCX Customer Found...");
String hash = payment.getTransactionHash();
custid = hcxPaybito.getCustomer_id();
pubKey = hcxPaybito.getHcx_pubkey();
privKey = hcxPaybito.getHcx_privkey();
hcx_balance = ((PaymentOperationResponse) payment).getAmount();
BigDecimal network_balance = new BigDecimal(hcx_balance);
System.out.println("Total HCX Balance " + network_balance);
System.out.println("Transfer Balance to DB without txn fee" + network_balance);
System.out.println("Transfer Balance to HCX user with txn fee"
+String.valueOf(network_balance.subtract(networktxn_fee)));
con = Utils.getConnection();
con.setAutoCommit(false);
boolean c = Utils.insertuser(custid, pubKey, privKey, network_balance, hash, con);
if (c == true) {
System.out.println("DB updated successfully");
try {
KeyPair source = KeyPair.fromSecretSeed(privKey);
KeyPair destination = KeyPair.fromAccountId(paybitohcx_user);
AccountResponse sourceAccount;
sourceAccount = server.accounts().account(source);
Transaction transaction = new Transaction.Builder(sourceAccount).addOperation(
```

```
new PaymentOperation.Builder(destination, new AssetTypeNative(),String.valueOf(
network_balance.subtract(networktxn_fee))).build()).addMemo(Memo.text("Fund
Transfer")).setTimeout(6000).build();
transaction.sign(source);
SubmitTransactionResponse response = server.submitTransaction(transaction);
if (response.isSuccess()) {
// txn_hash=response.getHash();
System.out.println("Successful Transfer!!!");
con.commit();
System.out.println("Network Update");
} else {
System.out.println(response.getExtras().getResultCodes().getTransactionResultCode());
System.out.println(response.getExtras().getResultCodes().getOperationsResultCodes());
con.rollback();
System.out.println("Network Error DB roll backed");
}
} catch (IOException e) {
con.rollback();
System.out.println("DB roll backed");
e.printStackTrace();
}
} else {
System.out.println("DB Not Updated Successfully..");
}
} catch (SQLException e) {
try {
con.rollback();
System.out.println("DB roll backed");
} catch (SQLException e1) {
e1.printStackTrace();
}
e.printStackTrace();
}
}
}
}
});
}
catch (Exception e) {
e.printStackTrace();
}
}
}
```

b) Exchange Project for HCX send and receive

Project Structure



data.properties file

hcx=Define tomcat url where HCNetService deployed for calling createAccount,fundTransfer and getBalance service.
USER= Define any username (ex: krish).
PASSWORD= Define any password (ex : krisPass).
userId= Define pool account CUSTOMER_ID (ex :101).
hcxurl=Define aurora url:443/accounts/

dispatcher-servlet.xml

```
<?xmlversion="1.0"encoding="UTF-8"?>
<beans:beansxmlns="http://www.springframework.org/schema/mvc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"xmlns:beans="http://www.springframework.org/schema/beans"
xmlns:context="http://www.springframework.org/schema/context"
xsi:schemaLocation="http://www.springframework.org/schema/mvc
http://www.springframework.org/schema/mvc/spring-mvc.xsd
http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
```

```
http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context.xsd">
<annotation-driven/>
<context:component-scanbase-package="com.hcx.adapter.api.controller"/>

<beans:beanclass="org.springframework.web.servlet.view.InternalResourceViewResolver">
<beans:propertyname="prefix"value="/WEB-INF/pages/"/>
<beans:propertyname="suffix"value=".jsp"/>
</beans:bean>

<beans:beanid="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
<beans:beanid="dataSource"class="org.springframework.jdbc.datasource.DriverManagerDa
taSource">
<beans:propertyname="driverClassName"value="oracle.jdbc.driver.OracleDriver"/>
<beans:propertyname="url"value="Define database url"/>
<beans:propertyname="username"value="Define database username"/>
<beans:propertyname="password"value="Define database password"/>
</beans:bean>

<!-- Configure to plugin JSON as request and response in method handler -->
<beans:beanclass="org.springframework.web.servlet.mvc.method.annotation.RequestMapp
ingHandlerAdapter">
<beans:propertyname="messageConverters">
<beans:list>
<beans:refbean="jsonMessageConverter"/>
</beans:list>
</beans:property>
</beans:bean>

<!-- Configure bean to convert JSON to POJO and vice versa -->
<beans:beanid="jsonMessageConverter"class="org.springframework.http.converter.json.Ma
ppingJackson2HttpMessageConverter">
</beans:bean>

<beans:beanid="WalletHistoryDao"class="com.hcx.adapter.dao.WalletHistoryDao">
<beans:propertyname="dataSource"ref="dataSource"/>
</beans:bean>

<beans:beanid="PaymentHistoryDao"class="com.hcx.adapter.dao.PaymentHistoryDao">
<beans:propertyname="dataSource"ref="dataSource"/>
</beans:bean>

<beans:beanid="CustomerLedgerDao"class="com.hcx.adapter.dao.CustomerLedgerDao">
<beans:propertyname="dataSource"ref="dataSource"/>
</beans:bean>
```

```
<beans:beanid="CustomerKeysDao"class="com.hcx.adapter.dao.CustomerKeysDao">
<beans:propertyname="dataSource"ref="dataSource"/>
</beans:bean>

<beans:beanid="WalletTo"class="com.hcx.adapter.dao.to.WalletTo"/>
<beans:beanid="walletHistTo"class="com.hcx.adapter.dao.to.WalletHistoryTo"/>
<beans:beanid="payRequestTo"class="com.hcx.adapter.dao.to.PaymentRequestTo"/>
<beans:beanid="paywithdrwalTo"class="com.hcx.adapter.dao.to.PaymentWithdrawlTo"/>
<beans:beanid="paymentHistTo"class="com.hcx.adapter.dao.to.PaymentHistoryTo"/>
<beans:beanid="customerLedgerTo"class="com.hcx.adapter.dao.to.CustomerLedgerTo"/>
<beans:beanid="customerKeysTo"class="com.hcx.adapter.dao.to.CustomerKeysTo"/>
<beans:beanid="walletService"class="com.hcx.component.service.WalletService">
<beans:propertyname="walletTo"ref="WalletTo"/>
</beans:bean>

<beans:beanid="walletHistoryService"class="com.btc.component.service.WalletHistoryServic
e">
<beans:propertyname="walletHistoryTo"ref="walletHistTo"/>
<beans:propertyname="walletHistoryDao"ref="WalletHistoryDao"/>
</beans:bean>

<beans:beanid="paymentRequestService"class="com.btc.component.service.PaymentReque
stService">
<beans:propertyname="paymentRequestTo"ref="payRequestTo"/>
<beans:propertyname="paymentRequestDao"ref="PayRequestDao"/>
</beans:bean>

<beans:beanid="paymentHistoryService"class="com.btc.component.service.PaymentHistory
Service">
<beans:propertyname="paymentHistoryTo"ref="paymentHistTo"/>
<beans:propertyname="paymentHistoryDao"ref="PaymentHistoryDao"/>
</beans:bean>
<beans:beanid="customerLedgerService"class="com.btc.component.service.CustomerLedger
Service">
<beans:propertyname="customerLedgerTo"ref="customerLedgerTo"/>
<beans:propertyname="customerLedgerDao"ref="CustomerLedgerDao"/>
</beans:bean>

<beans:beanid="customerKeyService"class="com.btc.component.service.CustomerKeyServic
e">
<beans:propertyname="customerKeysTo"ref="customerKeysTo"/>
<beans:propertyname="customerKeysDao"ref="CustomerKeysDao"/>
</beans:bean>
```

```
<beans:beanid="paybitoService"class="com.hcx.adapter.service.PayBitoWalletService">
<beans:propertyname="customerLedService"ref="customerLedgerService"/>
<beans:propertyname="paymentReqService"ref="paymentRequestService"/>
<beans:propertyname="paymentHistService"ref="paymentHistoryService"/>
<beans:propertyname="walletHistService"ref="walletHistoryService"/>
<beans:propertyname="paybitoWalletService"ref="walletService"/>
<beans:propertyname="customerKeyService"ref="customerKeyService"/>
</beans:bean>
</beans:beans>
```

i) How to generate HCX address for your exchange customer using createAccount service (receive)

Create model class for HCX send and rerecived address generate purpose:

PaymentProcessingRequestTo.java

```
package com.hcx.adapter.api.to;
publicclass PaymentProcessingRequestTo {
private String customer_Id;
private String toCustomer;
private String orderID;
private String description;
private String descriptionReceived;
private String status;
private String action;
private String fromadd;
private String toadd;
private String txnId;
private String txnCharge;
private String mining_fees;
private String currency;
private String transactionType;
private String baseCurrency;
private String creditAmount;
private String debitAmount;
private String offerPrice;
private String sendAmount;
```

Note:- Please create setter and getter method for all variables.

}

PaymentProcessingResponseTo.java

```
package com.hcx.adapter.api.to;
import com.fasterxml.jackson.annotation.JsonInclude;
@JsonInclude(JsonInclude.Include.NON_NULL)
publicclass PaymentProcessingResponseTo {
private String Customer_Id;
private String transaction_Id;
private String hcx_txnID;
public String description;
public String action;
private String Transaction_timestamp;
private String error;
private String publicKey;
private String txnCount;
private String txnCharge;
private String chargeSGST;
private String chargeCGST;
private String mining_fees;
private String currency;
private String debitHCXAmount;
private String creditHCXAmount;
private String closingHCXBalance;
private String requestAmount;
private String requestPrice;
private String status;
private String price;
```

Note:- Please create setter and getter method for all variables.

```
}
```

ClientResponse.java

```
package com.hcx.component.service;
publicclassClientResponse {
private String message;
private String hash;
private String code;
private String ledger;
private String destAccountPubKey;
private String destAccPrivKey;
private String statusCode;
private String balance;
private String amount;
private String fee;
private String source_account;
private String dest_acc;
```

```
private String created_at;
private String source_acc_seq;
```

Note:- Please create setter and getter method for all variables.

}

CustomerKeysTo.java

```
package com.hcx.adapter.dao.to;
publicclassCustomerKeysTo {
public String customerid;
public String fromadd;
public String toadd;
public String btcAmount;
public String txnId;
public String status;
public String action;
private String currency;
private String senderId;
private String senderPrivateKey;
private String senderPublicKey;
private String recvId;
private String receiverPrivateKey;
private String receiverPublicKey;
private String hcxLedger;
```

Note:- Please create setter and getter method for all variables.

}

PaymentHistoryTo.java

```
package com.hcx.adapter.dao.to;
import java.io.Serializable;
public class PaymentHistoryTo implements Serializable{
private static final long serialVersionUID = 1L;
private String transactionId;
private String customerId;
private String toCustomer;
private String orderID;
private String hcxTransactionId;
private String transactionTimestamp;
private String description;
private String action;
private String mining_fees;
```

```
private String currency;
private String debitHcxAmount;
private String creditHcxAmount;
private String closingHcxBalance;
private String status;
private String transactionType;
```

Note:- Please create setter and getter method for all variables.

```
}
```

HcxSendReceiveController.java(for HCX received address)

```
package com.hcx.adapter.api.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.api.to.PaymentProcessingResponseTo;
import com.hcx.adapter.service.HcxSendReceiveWalletService;

@RestController
@RequestMapping(value="/sendReceiveHcx")
public class HcxSendReceiveController {

@Autowired
HcxSendReceiveWalletService hcxService;

@RequestMapping(value = "/receiveHcx", method = RequestMethod.POST
,headers="Accept=application/json")
public ResponseEntity receiveFromOther(@RequestBody PaymentProcessingRequestTo
request) {

System.out.println("receive from other");
PaymentProcessingResponseTo response =  hcxService.receiveHcxFromOther(request);
return new ResponseEntity(response, HttpStatus.OK);

}

}
```

HcxSendReceiveWalletService.java

```
package com.hcx.adapter.service;
import com.hcx.adapter.api.to.CustomerRequestTo;
import com.hcx.adapter.api.to.CustomerResponseTo;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.api.to.PaymentProcessingResponseTo;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.adapter.dao.to.CustomerLedgerTo;
import com.hcx.adapter.dao.to.PaymentHistoryTo;
import com.hcx.client.api.to.WalletRequestTO;
import com.hcx.client.api.to.WalletResponseTO;
import com.hcx.client.api.walletapi.impl.IWalletRequestController;
import com.hcx.client.common.util.CommonUtil;
import com.hcx.component.service.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;
import java.net.URLConnection;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import javax.net.ssl.HttpsURLConnection;
import javax.swing.tree.TreeCellRenderer;
import org.apache.commons.io.IOUtils;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.springframework.beans.factory.annotation.Autowired;

public class HcxSendReceiveWalletService {
public CustomerLedgerService customerLedService;
public PaymentHistoryService paymentHistService;
public WalletHistoryService walletHistService;
public CustomerKeyService customerKeyService;
@Autowired
public HcxSendReceiveService payService;
public CustomerKeyService getCustomerKeyService() {
return customerKeyService;
}
public void setCustomerKeyService(CustomerKeyService customerKeyService) {
this.customerKeyService = customerKeyService;
```

```
}

public CustomerLedgerService getCustomerLedService() {
return customerLedService;
}

public void setCustomerLedService(CustomerLedgerService customerLedService) {
this.customerLedService = customerLedService;
}

public PaymentHistoryService getPaymentHistService() {
return paymentHistService;
}

public void setPaymentHistService(PaymentHistoryService paymentHistService) {
this.paymentHistService = paymentHistService;
}

public WalletHistoryService getWalletHistService() {
return walletHistService;
}
public void setWalletHistService(WalletHistoryService walletHistService) {
this.walletHistService = walletHistService;
}

public HcxSendReceiveService getPayService() {
return payService;
}


public PaymentProcessingResponseTo receiveHcxFromOther(PaymentProcessingRequestTo
paymentPReq){
PaymentProcessingResponseTo response=new PaymentProcessingResponseTo();
if(paymentPReq.getCurrency().equals("hcx")){
CustomerKeysTo
key=this.customerKeyService.getReceivingKey(paymentPReq.getCustomer_Id());
if(key.getReceiverPublicKey().equals("0")){
try {
ClientResponse res=(ClientResponse)new HcNetService().getCreateAccount();
if(res.getCode().equals("0")){
int
z=this.customerKeyService.updateHcxKeys(paymentPReq.getCustomer_Id(),res.getDestAccP
rivKey(),res.getDestAccountPubKey());
key.setReceiverPublicKey(res.getDestAccountPubKey());
key.setReceiverPrivateKey(res.getDestAccPrivKey());

}
```

```
} catch (IOException e) {
e.printStackTrace();
}
}
response.setPublicKey(key.getReceiverPublicKey());
}
}
```

CustomerKeyService.java

```
package com.hcx.component.service;

import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.dao.CustomerKeysDao;
import com.hcx.adapter.dao.to.CustomerKeysTo;

public class CustomerKeyService {
public CustomerKeysDao customerKeysDao;
public CustomerKeysTo customerKeysTo;

public CustomerKeysDao getCustomerKeysDao() {
return customerKeysDao;
}
public void setCustomerKeysDao(CustomerKeysDao customerKeysDao) {
this.customerKeysDao = customerKeysDao;
}
public CustomerKeysTo getCustomerKeysTo() {
return customerKeysTo;
}
public void setCustomerKeysTo(CustomerKeysTo customerKeysTo) {
this.customerKeysTo = customerKeysTo;
}

 public CustomerKeysTo getReceivingKey(String customer_Id) {
 return this.customerKeysDao.getreceiveAddressHcx(customer_Id);
}
}
```

CustomerKeysDao.java

```
package com.hcx.adapter.dao;
import java.util.List;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.client.common.util.CommonUtil;

public class CustomerKeysDao extends JdbcDaoSupport{

public CustomerKeysTo getreceiveAddressHcx(String customerKeysTo) {
String sql= "select nvl(HCX_PUBKEY, '0') as HCX_PUBKEY from HCX_BALANCE where
CUSTOMER_ID = '"+customerKeysTo+"'";
List<CustomerKeysTo> customerr =
(List<CustomerKeysTo>)this.getJdbcTemplate().query(sql,new
RowMapper<CustomerKeysTo>(){
@Override
public CustomerKeysTo mapRow(ResultSet rs,int rno)throws SQLException{
CustomerKeysTo customer=new CustomerKeysTo();
customer.setReceiverPublicKey(rs.getString("HCX_PUBKEY"));
return customer;
}
});
return customerr.get(0);
}
}
```

HcNetService.java

```
package com.hcx.component.service;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.hcx.client.common.util.CommonUtil;

public class HcNetService {
public  ClientResponse getCreateAccount() throws IOException  {
ClientResponse respons =null;
CommonUtil commonUtil = new CommonUtil();
String path = commonUtil.getProperty("hcx")+"createAccount";
```

```
System.out.println("URL :"+path);
HttpClient client = new DefaultHttpClient();
HttpPost post = new HttpPost(path);
StringEntity input = new StringEntity("{\"amount\":\"20\"}");
input.setContentType("application/json");
post.setEntity(input);
HttpResponse response = client.execute(post);
BufferedReader rd = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
String line = "";
ObjectMapper obj = new ObjectMapper();
while ((line = rd.readLine()) != null) {
respons = obj.readValue(line, ClientResponse.class);
}
return respons;
}
}
```

CommonUtil.java

```
package com.hcx.client.common.util;
import java.io.InputStream;
import java.util.Properties;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import java.util.Base64;
public class CommonUtil {
private static final Log logger = LogFactory.getLog(CommonUtil.class);
public String getProperty(String name) {
String propertyVal="";
Properties prop = new Properties();
try {
InputStream propertiesFile =
this.getClass().getClassLoader().getResourceAsStream("data.properties");
prop.load(propertiesFile);
propertyVal = prop.getProperty(name);
if(name.equalsIgnoreCase("USER")||name.equalsIgnoreCase("PASSWORD")){
byte[] base64decodedBytes = Base64.getDecoder().decode(propertyVal);
String strVal=new String(base64decodedBytes, "utf-8");
return strVal;
}else{
return propertyVal;
}
}
(Exception e) {
e.printStackTrace();
```

```
}
return propertyVal;
}
}
```

ii) How HCX is sent to same wallet (between your exchange customers) and other wallets (to customers of other exchanges)

a. HCX send to same wallet

HcxSendReceiveController.java

```
package com.hcx.adapter.api.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.api.to.PaymentProcessingResponseTo;
import com.hcx.adapter.service.HcxSendReceiveWalletService;

@RestController
@RequestMapping(value="/sendReceiveHcx")
public class HcxSendReceiveController {

@Autowired
HcxSendReceiveWalletService hcxService;

@RequestMapping(value = "/sendHcxSameWallet", method = RequestMethod.POST ,
headers="Accept=application/json")
publicResponseEntity customerSendHcxUser(@RequestBody PaymentProcessingRequestTo
request) {
System.out.println("sendHcxSameWallet");
PaymentProcessingResponseTo response =  hcxService.sendHcxToUser(request);
returnnew ResponseEntity(response, HttpStatus.OK);
}
}
```

HcxSendReceiveWalletService.java

```
package com.hcx.adapter.service;
import com.hcx.adapter.api.to.CustomerRequestTo;
import com.hcx.adapter.api.to.CustomerResponseTo;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.api.to.PaymentProcessingResponseTo;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.adapter.dao.to.CustomerLedgerTo;
import com.hcx.adapter.dao.to.PaymentHistoryTo;
import com.hcx.client.api.to.WalletRequestTO;
import com.hcx.client.api.to.WalletResponseTO;
import com.hcx.client.api.walletapi.impl.IWalletRequestController;
import com.hcx.client.common.util.CommonUtil;
import com.hcx.component.service.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;
import java.net.URLConnection;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import javax.net.ssl.HttpsURLConnection;
import javax.swing.tree.TreeCellRenderer;
import org.apache.commons.io.IOUtils;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.springframework.beans.factory.annotation.Autowired;

public class HcxSendReceiveWalletService {
public CustomerLedgerService customerLedService;
public PaymentHistoryService paymentHistService;
public WalletHistoryService walletHistService;
public CustomerKeyService customerKeyService;

@Autowired
public HcxSendReceiveService payService;
public CustomerKeyService getCustomerKeyService() {
return customerKeyService;
}
public void setCustomerKeyService(CustomerKeyService customerKeyService) {
```

```
this.customerKeyService = customerKeyService;
}
public CustomerLedgerService getCustomerLedService() {
return customerLedService;
}
public void setCustomerLedService(CustomerLedgerService customerLedService) {
this.customerLedService = customerLedService;
}
public PaymentHistoryService getPaymentHistService() {
return paymentHistService;
}
public void setPaymentHistService(PaymentHistoryService paymentHistService) {
this.paymentHistService = paymentHistService;
}
public WalletHistoryService getWalletHistService() {
return walletHistService;
}
public void setWalletHistService(WalletHistoryService walletHistService) {
this.walletHistService = walletHistService;
}

public PaymentProcessingResponseTo sendHcxToUser(PaymentProcessingRequestTo
paymentPReq){
paymentPReq.setTxnId(null);
PaymentProcessingResponseTo response = new PaymentProcessingResponseTo();
CustomerRequestTo req = new CustomerRequestTo();
req.setCustomer_Id(paymentPReq.getCustomer_Id());
CustomerResponseTo res = this.getCustomerBalance(req);
if(paymentPReq.getCurrency().equals("hcx")){
if(Double.parseDouble(res.getHcx_balance())>Double.parseDouble(paymentPReq.getBtc_a
mount())){
System.out.println(res.getHcx_balance());
ExecutorService executor = Executors.newFixedThreadPool(1);
HcNetSendThread sendStellarThread=new
HcNetSendThread(customerKeyService,walletHistService,paymentHistService,paymentPReq
);
executor.execute(sendStellarThread);
executor.shutdown();
while (!executor.isTerminated()) {
}
}else{
response.setError("Insufficient Fund ");
}
}
return response;
}
}
```

HcxSendReceiveWalletService.java

```
package com.hcx.adapter.service;
import com.hcx.adapter.api.to.CustomerRequestTo;
import com.hcx.adapter.api.to.CustomerResponseTo;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.api.to.PaymentProcessingResponseTo;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.adapter.dao.to.CustomerLedgerTo;
import com.hcx.adapter.dao.to.PaymentHistoryTo;
import com.hcx.client.api.to.WalletRequestTO;
import com.hcx.client.api.to.WalletResponseTO;
import com.hcx.client.api.walletapi.impl.IWalletRequestController;
import com.hcx.client.common.util.CommonUtil;
import com.hcx.component.service.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;
import java.net.URLConnection;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import javax.net.ssl.HttpsURLConnection;
import javax.swing.tree.TreeCellRenderer;
import org.apache.commons.io.IOUtils;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.springframework.beans.factory.annotation.Autowired;

public class HcxSendReceiveWalletService {
public CustomerLedgerService customerLedService;
public PaymentHistoryService paymentHistService;
public WalletHistoryService walletHistService;
public CustomerKeyService customerKeyService;

@Autowired
public HcxSendReceiveService payService;
public CustomerKeyService getCustomerKeyService() {
return customerKeyService;
}
```

```
public void setCustomerKeyService(CustomerKeyService customerKeyService) {
this.customerKeyService = customerKeyService;
}

public CustomerLedgerService getCustomerLedService() {
return customerLedService;
}

public void setCustomerLedService(CustomerLedgerService customerLedService) {
this.customerLedService = customerLedService;
}


public PaymentHistoryService getPaymentHistService() {
return paymentHistService;
}

public void setPaymentHistService(PaymentHistoryService paymentHistService) {
this.paymentHistService = paymentHistService;
}

public WalletHistoryService getWalletHistService() {
return walletHistService;
}

public void setWalletHistService(WalletHistoryService walletHistService) {
this.walletHistService = walletHistService;
}


public HcxSendReceiveService getPayService() {
return payService;
}

public synchronized CustomerResponseTo getCustomerBalance(CustomerRequestTo rar){
CustomerResponseTo response=new CustomerResponseTo();
CustomerLedgerTo customerLedger=customerLedService.customerLedgerList(rar);
response.setHcx_balance(customerLedger.getHcx_balance());
response.setBuyhcx(customerLedger.getBuyhcx());
response.setSellhcx(customerLedger.getSellhcx());
return response;
}
}
```

CustomerLedgerService.java

```
package com.hcx.component.service;
import java.util.List;
import com.hcx.adapter.api.to.CustomerRequestTo;
import com.hcx.adapter.dao.CustomerLedgerDao;
import com.hcx.adapter.dao.to.CustomerLedgerTo;
public class CustomerLedgerService {

public CustomerLedgerTo customerLedgerTo;
public CustomerLedgerDao customerLedgerDao;

public CustomerLedgerTo getCustomerLedgerTo() {
return customerLedgerTo;
}

public void setCustomerLedgerTo(CustomerLedgerTo customerLedgerTo) {
this.customerLedgerTo = customerLedgerTo;
}

public void setCustomerLedgerDao(CustomerLedgerDao customerLedgerDao) {
this.customerLedgerDao = customerLedgerDao;
}
public CustomerLedgerDao getCustomerLedgerDao() {
return customerLedgerDao;
}

public synchronized CustomerLedgerTo customerLedgerList(CustomerRequestTo rar){
customerLedgerTo.setCustomerId(rar.getCustomer_Id());
customerLedgerTo = customerLedgerDao.getCustomerLedger(customerLedgerTo);
return customerLedgerTo;
}
}
```

CustomerLedgerDao.java

```
package com.hcx.adapter.dao;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.ResultSetExtractor;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
```

```
import com.hcx.adapter.dao.to.CustomerLedgerTo;
import com.hcx.client.common.util.CommonUtil;

public class CustomerLedgerDao extends JdbcDaoSupport {
public synchronized CustomerLedgerTo getCustomerLedger(CustomerLedgerTo customer){
final String query = "select TransactionID, customer_ID, customer_name, "
+ "HCX_PRIVKEY,HCX_PUBKEY,HCX_BALANCE,BUY_HCX,SELL_HCX, "

+ "from hcx_balance WHERE customer_ID = ?";
return getJdbcTemplate().query(query , new PreparedStatementSetter() {

@Override
public void setValues(PreparedStatement arg0) throws SQLException {
arg0.setString(1, customer.getCustomerId());
}
}
, new ResultSetExtractor<List<CustomerLedgerTo>>() {
@Override
public List<CustomerLedgerTo> extractData(ResultSet rs) throws SQLException {
List<CustomerLedgerTo> lclt = new ArrayList<CustomerLedgerTo>();
if(rs.next()){
CustomerLedgerTo clt = new CustomerLedgerTo();
clt.setTransactionId(rs.getString("TransactionID"));
clt.setCustomerId(rs.getString("customer_ID"));
clt.setCustomerName(rs.getString("customer_name"));
clt.setHcx_privKey(rs.getString("HCX_PRIVKEY"));
clt.setHcx_pubKey(rs.getString("HCX_PUBKEY"));
clt.setHcx_balance(""+rs.getObject("HCX_BALANCE"));
clt.setBuyhcx(rs.getString("BUY_HCX"));
clt.setSellhcx(rs.getString("SELL_HCX"));
lclt.add(clt);
}
return lclt;
}}).get(0);
}
```

HcNetSendThread.java

```
package com.hcx.component.service;
import java.io.IOException;
import java.util.Base64;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.dao.to.CustomerKeysTo;
public class HcNetSendThread extends Thread {

public PaymentHistoryService paymentHistService;
public WalletHistoryService walletHistService;
```

```
public CustomerKeyService customerKeyService;
public PaymentProcessingRequestTo paymentPReq;

private String error;
public HcNetSendThread(CustomerKeyService customerKeyService, WalletHistoryService
walletHistService,
PaymentHistoryService paymentHistService, PaymentProcessingRequestTo paymentPReq) {
this.walletHistService=walletHistService;
this.paymentHistService=paymentHistService;
this.paymentPReq=paymentPReq;
this.customerKeyService=customerKeyService;
}
public void run(){
sendAndReceiveCalling();
}

private void sendAndReceiveCalling() {
if(paymentPReq.getToCustomer()!=null){
CustomerKeysTo
keys=this.customerKeyService.getSenderAndReceiverKeys("101",paymentPReq.getToCusto
mer());
if(keys.getReceiverPublicKey().equals("0")){
try {
ClientResponse response=(ClientResponse)new HcNetService().getCreateAccount();
if(response.getCode().equals("0")){
int
z=this.customerKeyService.updateHcxKeys(paymentPReq.getToCustomer(),response.getDes
tAccPrivKey(),response.getDestAccountPubKey());
keys.setReceiverPublicKey(response.getDestAccountPubKey());
keys.setReceiverPrivateKey(response.getDestAccPrivKey());
}
} catch (IOException e) {
e.printStackTrace();
}
}
ClientResponse response=null;
try {
response = new HcNetService().betweenTwoUsersHCX(keys.getSenderPrivateKey(),
keys.getReceiverPublicKey(), paymentPReq.getBtc_amount());
} catch (IOException e) {
e.printStackTrace();
}
if(response.getCode().equals("0")){
paymentPReq.setFromadd(keys.getSenderPrivateKey());
paymentPReq.setToadd(keys.getReceiverPublicKey());
paymentPReq.setTxnId(response.getHash());
paymentPReq.setCurrency("hcx");
```

```
paymentPReq.setStatus("send");
paymentPReq.setTxnId(response.getHash());
paymentPReq.setCurrency("hcx");
paymentPReq.setStatus("sending to paybito");
paymentPReq.setMining_fees("0.00001");
paymentPReq.setBtc_amount(paymentPReq.getBtc_amount());
System.out.println("HCX Amount::"+paymentPReq.getBtc_amount());
int x =  customerKeyService.insertCustomerKeys(paymentPReq,response.getLedger());
System.out.println("customer key :"+ x);
paymentHistService.insertSendPaymentHistory(paymentPReq);
}
CustomerKeysTo keys=this.customerKeyService.getSenderPrivateKey("101");
System.out.println("Sender Private Key::"+keys.getSenderPrivateKey());
ClientResponse responsee=null;
try {
responsee = new HcNetService().betweenTwoUsersHCX(keys.getSenderPrivateKey(),
paymentPReq.getToadd(), paymentPReq.getBtc_amount());
} catch (IOException e) {
e.printStackTrace();
}
if(responsee.getCode().equals("0")){
paymentPReq.setFromadd(keys.getSenderPrivateKey());
System.out.println("From Address:::"+paymentPReq.getFromadd());
paymentPReq.setTxnId(responsee.getHash());
paymentPReq.setCurrency("hcx");
paymentPReq.setStatus("sending other");
paymentPReq.setMining_fees("0.00001");
paymentPReq.setBtc_amount(paymentPReq.getBtc_amount());
paymentPReq.setDescriptionReceived("HCX Received");
int x =  customerKeyService.insertCustomerKeys(paymentPReq,responsee.getLedger());
System.out.println("customer key :"+ x);
paymentHistService.insertSendPaymentHistory(paymentPReq);
}else{
this.setError("HCX not sent. Please verify receiving address and transaction amount");
}
}
}
public String getError() {
return error;
}
public void setError(String error) {
this.error = error;
}
}
```

CustomerKeyService.java

```
package com.hcx.component.service;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.dao.CustomerKeysDao;
import com.hcx.adapter.dao.to.CustomerKeysTo;
public class CustomerKeyService {
public CustomerKeysDao customerKeysDao;
public CustomerKeysTo customerKeysTo;

public CustomerKeysDao getCustomerKeysDao() {
return customerKeysDao;}
public void setCustomerKeysDao(CustomerKeysDao customerKeysDao) {
this.customerKeysDao = customerKeysDao;
}
public CustomerKeysTo getCustomerKeysTo() {
return customerKeysTo;
}
public void setCustomerKeysTo(CustomerKeysTo customerKeysTo) {
this.customerKeysTo = customerKeysTo;
}

public synchronized CustomerKeysTo getSenderAndReceiverKeys(String fromId, String toId)
{
return this.customerKeysDao.getKeysSenderAndReceiver(fromId,toId);
}
```

CustomerKeysDao.java

```
package com.hcx.adapter.dao;
import java.util.List;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.client.common.util.CommonUtil;

public class CustomerKeysDao extends JdbcDaoSupport{
public synchronized CustomerKeysTo getKeysSenderAndReceiver(String fromId, String toID)
{

String sql= "select c1.HCX_PRIVKEY as sendPrivateKey, c1.HCX_PUBKEY as sendPrivtaeKey,
nvl(c2.HCX_PRIVKEY,'0') as recevPrivateKey, nvl(c2.HCX_PUBKEY, '0') as recevPublicKey "+
```

```
"from HCX_BALANCE c1, HCX_BALANCEc2 where  c1.customer_ID!=c2.customer_id and
c1.customer_id='"+fromId+"' and c2.customer_id='"+toID+"'";
System.out.println(sql);

List<CustomerKeysTo> customerr =  (List<CustomerKeysTo>)this.getJdbcTemplate().query(
sql,new RowMapper<CustomerKeysTo>(){
@Override
public CustomerKeysTo mapRow(ResultSet rs,int rno)throws SQLException{
CustomerKeysTo customer=new CustomerKeysTo();
customer.setSenderPrivateKey(rs.getString("sendPrivateKey"));
customer.setSenderPublicKey(rs.getString("sendPrivtaeKey"));
customer.setReceiverPrivateKey(rs.getString("recevPrivateKey"));
customer.setReceiverPublicKey(rs.getString("recevPublicKey"));
return customer;
}
});
return customerr.get(0);
}

}
```

HcNetService.java

```
package com.hcx.component.service;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.hcx.client.common.util.CommonUtil;

public class HcNetService {
public  ClientResponse getCreateAccount() throws IOException  {
ClientResponse respons =null;
CommonUtil commonUtil = new CommonUtil();
String path = commonUtil.getProperty("hcx")+"createAccount";
System.out.println("URL :"+path);
HttpClient client = new DefaultHttpClient();
HttpPost post = new HttpPost(path);
StringEntity input = new StringEntity("{\"amount\":\"20\"}");
input.setContentType("application/json");
```

```
post.setEntity(input);
HttpResponse response = client.execute(post);
BufferedReader rd = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
String line = "";
ObjectMapper obj = new ObjectMapper();
while ((line = rd.readLine()) != null) {
respons = obj.readValue(line, ClientResponse.class);
}
return respons;
}
}
```

CommonUtil.java

```
package com.hcx.client.common.util;
import java.io.InputStream;
import java.util.Properties;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import java.util.Base64;
public class CommonUtil {
private static final Log logger = LogFactory.getLog(CommonUtil.class);
public String getProperty(String name) {
String propertyVal="";
Properties prop = new Properties();
try {
InputStream propertiesFile =
this.getClass().getClassLoader().getResourceAsStream("data.properties");
prop.load(propertiesFile);
propertyVal = prop.getProperty(name);
if(name.equalsIgnoreCase("USER")||name.equalsIgnoreCase("PASSWORD")){
byte[] base64decodedBytes = Base64.getDecoder().decode(propertyVal);
String strVal=new String(base64decodedBytes, "utf-8");
return strVal;
}else{
return propertyVal;
}
}
(Exception e) {
e.printStackTrace();
}
return propertyVal;
}
}
```

CustomerKeyService.java

```
package com.hcx.component.service;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.dao.CustomerKeysDao;
import com.hcx.adapter.dao.to.CustomerKeysTo;
public class CustomerKeyService {
public CustomerKeysDao customerKeysDao;
public CustomerKeysTo customerKeysTo;

public CustomerKeysDao getCustomerKeysDao() {
return customerKeysDao;}
public void setCustomerKeysDao(CustomerKeysDao customerKeysDao) {
this.customerKeysDao = customerKeysDao;
}
public CustomerKeysTo getCustomerKeysTo() {
return customerKeysTo;
}
public void setCustomerKeysTo(CustomerKeysTo customerKeysTo) {
this.customerKeysTo = customerKeysTo;
}

publicsynchronizedintupdateHcxKeys(String customer_Id, String accPrivKey, String
accPubKey) {
returnthis.customerKeysDao.updateHcxKeysCustomer(customer_Id,accPrivKey,accPubKey);
}
}
```

CustomerKeysDao.java

```
package com.hcx.adapter.dao;
import java.util.List;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.client.common.util.CommonUtil;

public class CustomerKeysDao extends JdbcDaoSupport{

publicsynchronizedintupdateHcxKeysCustomer(String customer_Id,String privKey,String
pubKey) {
String query = "UPDATE HCX_BALANCE  SET HCX_PUBKEY ='"+pubKey+"', HCX_PRIVKEY
='"+privKey+"', "
```

```
+ "BUY_HCX=0.0, SELL_HCX=0.0, HCX_BALANCE =0.0  WHERE customer_id
='"+customer_Id+"'";
intcount=this.getJdbcTemplate().update(query);
returncount;
}
}
```

HcNetService.java

```
package com.hcx.component.service;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.hcx.client.common.util.CommonUtil;

public class HcNetService {
public  ClientResponse betweenTwoUsersHCX(String fromPrivateKey, String toPublicKey,
String strAmount) throws IOException {
ClientResponse respons=null;
CommonUtil commonUtil = new CommonUtil();
String path = commonUtil.getProperty("hcx")+"fundTransfer";
HttpClient client = newDefaultHttpClient();
HttpPost post = new HttpPost(path);
StringEntity input = new
StringEntity("{\"fromSecretSeed\":\""+fromPrivateKey+"\",\"destination\":\""+toPublicKey+
"\",\"amount\":\""+strAmount+"\"}");
input.setContentType("application/json");
post.setEntity(input);
HttpResponse response = client.execute(post);
BufferedReader rd = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
String line = "";
ObjectMapper obj = new ObjectMapper();
while ((line = rd.readLine()) != null) {
respons = obj.readValue(line, ClientResponse.class);
}
returnrespons;
}
}
```

CustomerKeyService.java

```
package com.hcx.component.service;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.dao.CustomerKeysDao;
import com.hcx.adapter.dao.to.CustomerKeysTo;
public class CustomerKeyService {
public CustomerKeysDao customerKeysDao;
public CustomerKeysTo customerKeysTo;

public CustomerKeysDao getCustomerKeysDao() {
return customerKeysDao;}
public void setCustomerKeysDao(CustomerKeysDao customerKeysDao) {
this.customerKeysDao = customerKeysDao;
}
public CustomerKeysTo getCustomerKeysTo() {
return customerKeysTo;
}
public void setCustomerKeysTo(CustomerKeysTo customerKeysTo) {
this.customerKeysTo = customerKeysTo;
}

publicsynchronizedintinsertCustomerKeys(PaymentProcessingRequestTo pprt, String
strLedger){
customerKeysTo.setCustomerid(pprt.getCustomer_Id());
customerKeysTo.setFromadd(pprt.getFromadd());
customerKeysTo.setToadd(pprt.getToadd());
customerKeysTo.setBtcAmount(pprt.getBtc_amount());
customerKeysTo.setTxnId(pprt.getTxnId());
customerKeysTo.setStatus(pprt.getStatus());
customerKeysTo.setAction(pprt.getAction());
customerKeysTo.setCurrency(pprt.getCurrency());
intx = customerKeysDao.insertCustomerKeyDao(customerKeysTo,strLedger);
returnx;
}
}
```

CustomerKeysDao.java

```
package com.hcx.adapter.dao;
import java.util.List;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
```

```
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.client.common.util.CommonUtil;

public class CustomerKeysDao extends JdbcDaoSupport{

publicintinsertCustomerKeyDao(CustomerKeysTo customerKeysTo, String ledger_no) {
Long tran_id = (Long)this.getJdbcTemplate().query("select coalesce(max(ID),0)+1 from
CUSTOMERHCXKEYS",new SingleRowExtractor(SingleRowExtractor.LONG));
String query="insert into CUSTOMERHCXKEYS (id,customerid,
fromadd,toadd,hcxamount,txnId,status,action,currency,ledger) values(?,?,?,?,?,?,?,?,?,?)";
intx=this.getJdbcTemplate().update(query,new PreparedStatementSetter() {
@Override
publicvoid setValues(PreparedStatement ps) throws SQLException {
inti = 0;
ps.setObject(++i, tran_id);
ps.setObject(++i,customerKeysTo.getCustomerid()!=null?customerKeysTo.getCustomerid():
null);
ps.setObject(++i, customerKeysTo.getFromadd()!=null?customerKeysTo.getFromadd():null);
ps.setObject(++i, customerKeysTo.getToadd()!=null?customerKeysTo.getToadd():null);
ps.setObject(++i,customerKeysTo.getBtcAmount()!=null?customerKeysTo.getBtcAmount():"
0.0");
ps.setObject(++i, customerKeysTo.getTxnId()!=null? customerKeysTo.getTxnId(): null);
ps.setObject(++i, customerKeysTo.getStatus()!=null?customerKeysTo.getStatus():null);
ps.setObject(++i, customerKeysTo.getAction()!=null?customerKeysTo.getAction():null);
ps.setObject(++i, customerKeysTo.getCurrency()!=null?customerKeysTo.getCurrency():null);
ps.setObject(++i, ledger_no);
}
});
returnx;
}
}
```

PaymentHistoryService.java

```
package com.hcx.component.service;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import com.hcx.adapter.api.to.CustomerRequestTo;
import com.hcx.adapter.api.to.CustomerResponseTo;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.api.to.PaymentProcessingResponseTo;
import com.hcx.adapter.dao.PaymentHistoryDao;
import com.hcx.adapter.dao.to.PaymentHistoryTo;
public class PaymentHistoryService {
public PaymentHistoryTo paymentHistoryTo;
public PaymentHistoryDao paymentHistoryDao;
```

```
public PaymentHistoryTo getPaymentHistoryTo() {
return paymentHistoryTo;
}
public void setPaymentHistoryTo(PaymentHistoryTo paymentHistoryTo) {
this.paymentHistoryTo = paymentHistoryTo;
}
public void setPaymentHistoryDao(PaymentHistoryDao paymentHistoryDao) {
this.paymentHistoryDao = paymentHistoryDao;
}
public PaymentHistoryDao getPaymentHistoryDao() {
return paymentHistoryDao;
}

public synchronized int insertSendPaymentHistory(PaymentProcessingRequestTo rar){
paymentHistoryTo.setCreditHcxAmount(null);
paymentHistoryTo.setDebitHcxAmount(null);
paymentHistoryTo.setHcxTransactionId(null);
if(rar.getCurrency().equals("hcx")){
paymentHistoryTo.setCreditHcxAmount(null);
paymentHistoryTo.setDebitHcxAmount(rar.getSendamount());
paymentHistoryTo.setHcxTransactionId(rar.getTxnId());
}

paymentHistoryTo.setCustomerId(rar.getCustomer_Id());
paymentHistoryTo.setDescription(rar.getDescription());
paymentHistoryTo.setAction(rar.getAction());
paymentHistoryTo.setTxnCharge(null);
paymentHistoryTo.setChargeCGST(null);
paymentHistoryTo.setChargeSGST(null);
paymentHistoryTo.setMining_fees(rar.getMining_fees());
paymentHistoryTo.setCurrency(rar.getCurrency());
paymentHistoryTo.setStatus("0");
paymentHistoryTo.setOrderID(null);
int x = paymentHistoryDao.insertPaymentHistory(paymentHistoryTo);
return x;
}
}
```

PaymentHistoryDao.java

```
package com.hcx.adapter.dao;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.util.List;
import java.util.Map;
```

```
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.BatchPreparedStatementSetter;
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.ResultSetExtractor;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.SqlParameterSource;
import org.springframework.jdbc.core.simple.SimpleJdbcCall;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import com.hcx.adapter.dao.to.CustomerLedgerTo;
import com.hcx.adapter.dao.to.PaymentHistoryTo;
import com.hcx.client.common.util.CommonUtil;

public class PaymentHistoryDao extends JdbcDaoSupport {
public synchronized int insertPaymentHistory(PaymentHistoryTo payment) {
Long tran_id = (Long)this.getJdbcTemplate().query("select coalesce(max(TransactionID),0)+1
from payment_history",new SingleRowExtractor(SingleRowExtractor.LONG));
final Timestamp timestamp = (Timestamp)this.getJdbcTemplate().query("select
current_timestamp from dual",new SingleRowExtractor(SingleRowExtractor.TIMESTAMP));
payment.setTransactionId(tran_id.toString());

String sql="insert into payment_history (TransactionID, customer_ID, orderid, "
+ "txnCharge, sgst, cgst, miningFees, "
+ "transaction_timestamp, description, action, "
+ "hcx_txnID,DEBIT_HCX_AMOUNT,CREDIT_HCX_AMOUNT,currency,
BASECURRENCY,status,networkFees) "
+ "values ( ?, ?, ? ,?, ?, ?, ? ,?, ?, ?,?,?,?,?, ?,?,? )";

int x=this.getJdbcTemplate().update(sql,new PreparedStatementSetter() {

@Override
public void setValues(PreparedStatement ps) throws SQLException {
int i = 0;
ps.setObject(++i,payment.getTransactionId()) ;
ps.setString(++i, payment.getCustomerId());
ps.setString(++i, payment.getOrderID()!=null?payment.getOrderID():null);
ps.setObject(++i, payment.getTxnCharge()!=null ? payment.getTxnCharge() :"0.0");
ps.setObject(++i, payment.getChargeSGST()!=null ? payment.getChargeSGST() :"0.0");
ps.setObject(++i, payment.getChargeCGST()!=null ? payment.getChargeCGST() :"0.0");
ps.setObject(++i, payment.getMining_fees()!=null ? payment.getMining_fees() :"0.0");
ps.setTimestamp(++i, timestamp);
ps.setString(++i, payment.getDescription()!=null? payment.getDescription():"oops");
ps.setString(++i, payment.getAction()!=null?payment.getAction():"pending");
ps.setString(++i, payment.getHcxTransactionId()!=null?payment.getHcxTransactionId():"-");
ps.setString(++i,
payment.getDebitHcxAmount()!=null?payment.getDebitHcxAmount():"0.0");
```

```
ps.setString(++i,
payment.getCreditHcxAmount()!=null?payment.getCreditHcxAmount():"0.0");
ps.setString(++i, payment.getCurrency()!=null?payment.getCurrency():null);
ps.setString(++i, payment.getBaseCurrency()!=null?payment.getBaseCurrency():null);
ps.setString(++i, payment.getStatus()!=null? payment.getStatus():"0");
ps.setString(++i, payment.getNetworkFees()!=null? payment.getNetworkFees():"0.0");
}
});
this.callingProcedure(payment, tran_id);
return x;

}
}
```

CustomerKeyService.java

```
package com.hcx.component.service;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.dao.CustomerKeysDao;
import com.hcx.adapter.dao.to.CustomerKeysTo;
public class CustomerKeyService {
public CustomerKeysDao customerKeysDao;
public CustomerKeysTo customerKeysTo;

public CustomerKeysDao getCustomerKeysDao() {
return customerKeysDao;}
public void setCustomerKeysDao(CustomerKeysDao customerKeysDao) {
this.customerKeysDao = customerKeysDao;
}
public CustomerKeysTo getCustomerKeysTo() {
return customerKeysTo;
}
public void setCustomerKeysTo(CustomerKeysTo customerKeysTo) {
this.customerKeysTo = customerKeysTo;
}

publicsynchronized CustomerKeysTo getSenderPrivateKey(String customer_Id) {
returnthis.customerKeysDao.getKeysSender(customer_Id);
}
}
```

CustomerKeysDao.java

```
package com.hcx.adapter.dao;
import java.util.List;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
```

```
import java.sql.SQLException;
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.client.common.util.CommonUtil;

public class CustomerKeysDao extends JdbcDaoSupport{

publicsynchronized CustomerKeysTo getKeysSender(String customer_Id) {

String sql= "select HCX_PRIVKEY  as sendPrivateKey from HCX_BALANCE where customer_id
= '"+customer_Id+"'";
List<CustomerKeysTo>customerr =  (List<CustomerKeysTo>)this.getJdbcTemplate().query(
sql,new RowMapper<CustomerKeysTo>(){
@Override
public CustomerKeysTo mapRow(ResultSet rs,intrno)throws SQLException{
CustomerKeysTocustomer=new CustomerKeysTo();
customer.setSenderPrivateKey(rs.getString("sendPrivateKey"));
returncustomer;
}
});
returncustomerr.get(0);
}
}
```

b. *HCX send to other wallet*

HcxSendReceiveController.java

```
package com.hcx.adapter.api.controller;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.bind.annotation.RestController;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.api.to.PaymentProcessingResponseTo;
import com.hcx.adapter.service.HcxSendReceiveWalletService;

@RestController
@RequestMapping(value="/sendReceiveHcx")
public class HcxSendReceiveController {
```

```
@Autowired
HcxSendReceiveWalletService hcxService;

@RequestMapping(value = "/sendHcxToOther", method = RequestMethod.POST ,
headers="Accept=application/json")
publicResponseEntity sendHcxoOther(@RequestBody PaymentProcessingRequestTo
request) {
System.out.println("sendHcxToOther");
PaymentProcessingResponseTo response =  hcxService.sendHcxToOther(request);
returnnew ResponseEntity(response, HttpStatus.OK);

}
}
```

HcxSendReceiveWalletService.java

```
package com.hcx.adapter.service;
import com.hcx.adapter.api.to.CustomerRequestTo;
import com.hcx.adapter.api.to.CustomerResponseTo;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.api.to.PaymentProcessingResponseTo;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.adapter.dao.to.CustomerLedgerTo;
import com.hcx.adapter.dao.to.PaymentHistoryTo;
import com.hcx.client.api.to.WalletRequestTO;
import com.hcx.client.api.to.WalletResponseTO;
import com.hcx.client.api.walletapi.impl.IWalletRequestController;
import com.hcx.client.common.util.CommonUtil;
import com.hcx.component.service.*;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;
import java.net.URLConnection;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import javax.net.ssl.HttpsURLConnection;
import javax.swing.tree.TreeCellRenderer;
import org.apache.commons.io.IOUtils;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
```

```
import org.springframework.beans.factory.annotation.Autowired;

public class HcxSendReceiveWalletService {

public CustomerLedgerService customerLedService;
public PaymentHistoryService paymentHistService;
public WalletHistoryService walletHistService;
public CustomerKeyService customerKeyService;

@Autowired
public HcxSendReceiveService payService;
public CustomerKeyService getCustomerKeyService() {
return customerKeyService;
}
public void setCustomerKeyService(CustomerKeyService customerKeyService) {
this.customerKeyService = customerKeyService;
}
public CustomerLedgerService getCustomerLedService() {
return customerLedService;
}
public void setCustomerLedService(CustomerLedgerService customerLedService) {
this.customerLedService = customerLedService;
}
public PaymentHistoryService getPaymentHistService() {
return paymentHistService;
}
public void setPaymentHistService(PaymentHistoryService paymentHistService) {
this.paymentHistService = paymentHistService;
}
public WalletHistoryService getWalletHistService() {
return walletHistService;
}
public void setWalletHistService(WalletHistoryService walletHistService) {
this.walletHistService = walletHistService;
}

public PaymentProcessingResponseTo sendHcxToOther(PaymentProcessingRequestTo
paymentPReq){
String strError=null;
PaymentProcessingResponseTo response=new PaymentProcessingResponseTo();
CustomerRequestTo req=new CustomerRequestTo();
req.setCustomer_Id(paymentPReq.getCustomer_Id());
CustomerResponseTo res=this.getCustomerBalance(req);
if(paymentPReq.getCurrency().equals("hcx")){

if(Double.parseDouble(res.getHcx_balance())>Double.parseDouble(paymentPReq.getBtc_a
mount())){
```

```
System.out.println(res.getHcx_balance());
try {
if(checkHcxKeys(paymentPReq)==false){
ExecutorService executor = Executors.newFixedThreadPool(1);
HcNetSendThread sendStellarThread=new
HcNetSendThread(customerKeyService,walletHistService,paymentHistService,paymentPReq
);
executor.execute(sendStellarThread);
executor.shutdown();
while (!executor.isTerminated()) {}
response=this.getClosingBalance(paymentPReq);
returnresponse;
}else{
response.setError("HCX not sent. Please verify receiving address and transaction amount");
}
} catch (Exception e) {
e.printStackTrace();
}
}else{
response.setError("Insufficient Fund ");
}
}
returnresponse;
}
}
```

HcxSendReceiveWalletService.java

```
package com.hcx.adapter.service;

import com.hcx.adapter.api.to.CustomerRequestTo;
import com.hcx.adapter.api.to.CustomerResponseTo;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.api.to.PaymentProcessingResponseTo;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.adapter.dao.to.CustomerLedgerTo;
import com.hcx.adapter.dao.to.PaymentHistoryTo;

import com.hcx.client.api.to.WalletRequestTO;
import com.hcx.client.api.to.WalletResponseTO;
import com.hcx.client.api.walletapi.impl.IWalletRequestController;
import com.hcx.client.common.util.CommonUtil;
import com.hcx.component.service.*;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStream;
```

```
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.ProtocolException;
import java.net.URL;
import java.net.URLConnection;
import java.util.List;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;

import javax.net.ssl.HttpsURLConnection;
import javax.swing.tree.TreeCellRenderer;

import org.apache.commons.io.IOUtils;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.springframework.beans.factory.annotation.Autowired;

public class HcxSendReceiveWalletService {
public CustomerLedgerService customerLedService;
public PaymentHistoryService paymentHistService;
public WalletHistoryService walletHistService;
public CustomerKeyService customerKeyService;

@Autowired
public HcxSendReceiveService payService;
public CustomerKeyService getCustomerKeyService() {
return customerKeyService;
}

public void setCustomerKeyService(CustomerKeyService customerKeyService) {
this.customerKeyService = customerKeyService;
}

public CustomerLedgerService getCustomerLedService() {
return customerLedService;
}

public void setCustomerLedService(CustomerLedgerService customerLedService) {
this.customerLedService = customerLedService;
}


public PaymentHistoryService getPaymentHistService() {
return paymentHistService;
}
```

```
public void setPaymentHistService(PaymentHistoryService paymentHistService) {
this.paymentHistService = paymentHistService;
}

public WalletHistoryService getWalletHistService() {
return walletHistService;
}

public void setWalletHistService(WalletHistoryService walletHistService) {
this.walletHistService = walletHistService;
}

public synchronized CustomerResponseTo getCustomerBalance(CustomerRequestTo rar){
CustomerResponseTo response=new CustomerResponseTo();
CustomerLedgerTo customerLedger=customerLedService.customerLedgerList(rar);
response.setHcx_balance(customerLedger.getHcx_balance());
response.setBuyhcx(customerLedger.getBuyhcx());
response.setSellhcx(customerLedger.getSellhcx());
return response;
}
}
```

CustomerLedgerService.java

```
package com.hcx.component.service;
import java.util.List;
import com.hcx.adapter.api.to.CustomerRequestTo;
import com.hcx.adapter.dao.CustomerLedgerDao;
import com.hcx.adapter.dao.to.CustomerLedgerTo;
public class CustomerLedgerService {

public CustomerLedgerTo customerLedgerTo;
public CustomerLedgerDao customerLedgerDao;

public CustomerLedgerTo getCustomerLedgerTo() {
return customerLedgerTo;
}

public void setCustomerLedgerTo(CustomerLedgerTo customerLedgerTo) {
this.customerLedgerTo = customerLedgerTo;
}

public void setCustomerLedgerDao(CustomerLedgerDao customerLedgerDao) {
this.customerLedgerDao = customerLedgerDao;
}
public CustomerLedgerDao getCustomerLedgerDao() {
```

```
return customerLedgerDao;
}

public synchronized CustomerLedgerTo customerLedgerList(CustomerRequestTo rar){
customerLedgerTo.setCustomerId(rar.getCustomer_Id());
customerLedgerTo = customerLedgerDao.getCustomerLedger(customerLedgerTo);
return customerLedgerTo;
}
}
```

CustomerLedgerDao.java

```
package com.hcx.adapter.dao;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.ResultSetExtractor;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import com.hcx.adapter.dao.to.CustomerLedgerTo;
import com.hcx.client.common.util.CommonUtil;

public class CustomerLedgerDao extends JdbcDaoSupport {
public synchronized CustomerLedgerTo getCustomerLedger(CustomerLedgerTo customer){
final String query = "select TransactionID, customer_ID, customer_name, "
+ "HCX_PRIVKEY,HCX_PUBKEY,HCX_BALANCE,BUY_HCX,SELL_HCX, "

+ "from hcx_balance WHERE customer_ID = ?";
return getJdbcTemplate().query(query , new PreparedStatementSetter() {

@Override
public void setValues(PreparedStatement arg0) throws SQLException {
arg0.setString(1, customer.getCustomerId());
}
}
, new ResultSetExtractor<List<CustomerLedgerTo>>() {
@Override
public List<CustomerLedgerTo> extractData(ResultSet rs) throws SQLException {
List<CustomerLedgerTo> lclt = new ArrayList<CustomerLedgerTo>();
if(rs.next()){
CustomerLedgerTo clt = new CustomerLedgerTo();
clt.setTransactionId(rs.getString("TransactionID"));
clt.setCustomerId(rs.getString("customer_ID"));
```

```
clt.setCustomerName(rs.getString("customer_name"));
clt.setHcx_privKey(rs.getString("HCX_PRIVKEY"));
clt.setHcx_pubKey(rs.getString("HCX_PUBKEY"));
clt.setHcx_balance(""+rs.getObject("HCX_BALANCE"));
clt.setBuyhcx(rs.getString("BUY_HCX"));
clt.setSellhcx(rs.getString("SELL_HCX"));
lclt.add(clt);
}
return lclt;
}}).get(0);
}
```

HcNetSendThread.java

```
package com.hcx.component.service;
import java.io.IOException;
import java.util.Base64;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.dao.to.CustomerKeysTo;
public class HcNetSendThread extends Thread {

public PaymentHistoryService paymentHistService;
public WalletHistoryService walletHistService;
public CustomerKeyService customerKeyService;
public PaymentProcessingRequestTo paymentPReq;

private String error;
public HcNetSendThread(CustomerKeyService customerKeyService, WalletHistoryService
walletHistService,
PaymentHistoryService paymentHistService, PaymentProcessingRequestTo paymentPReq) {
this.walletHistService=walletHistService;
this.paymentHistService=paymentHistService;
this.paymentPReq=paymentPReq;
this.customerKeyService=customerKeyService;
}
public void run(){
sendAndReceiveCalling();
}

private void sendAndReceiveCalling() {

if(paymentPReq.getToCustomer()!=null){
CustomerKeysTo
keys=this.customerKeyService.getSenderAndReceiverKeys("101",paymentPReq.getToCusto
mer());
if(keys.getReceiverPublicKey().equals("0")){
try {
```

```
ClientResponse response=(ClientResponse)new HcNetService().getCreateAccount();
if(response.getCode().equals("0")){
int
z=this.customerKeyService.updateHcxKeys(paymentPReq.getToCustomer(),response.getDes
tAccPrivKey(),response.getDestAccountPubKey());
keys.setReceiverPublicKey(response.getDestAccountPubKey());
keys.setReceiverPrivateKey(response.getDestAccPrivKey());
}
} catch (IOException e) {
e.printStackTrace();
}
}
ClientResponse response=null;
try {
response = new HcNetService().betweenTwoUsersHCX(keys.getSenderPrivateKey(),
keys.getReceiverPublicKey(), paymentPReq.getBtc_amount());
} catch (IOException e) {
e.printStackTrace();
}
if(response.getCode().equals("0")){
paymentPReq.setFromadd(keys.getSenderPrivateKey());
paymentPReq.setToadd(keys.getReceiverPublicKey());
paymentPReq.setTxnId(response.getHash());
paymentPReq.setCurrency("hcx");
paymentPReq.setStatus("send");
paymentPReq.setTxnId(response.getHash());
paymentPReq.setCurrency("hcx");
paymentPReq.setStatus("sending to paybito");
paymentPReq.setMining_fees("0.00001");
paymentPReq.setBtc_amount(paymentPReq.getBtc_amount());
System.out.println("HCX Amount::"+paymentPReq.getBtc_amount());
int x =  customerKeyService.insertCustomerKeys(paymentPReq,response.getLedger());
System.out.println("customer key :"+ x);
paymentHistService.insertSendPaymentHistory(paymentPReq);
}
CustomerKeysTo keys=this.customerKeyService.getSenderPrivateKey("101");
System.out.println("Sender Private Key::"+keys.getSenderPrivateKey());
ClientResponse responsee=null;
try {
responsee = new HcNetService().betweenTwoUsersHCX(keys.getSenderPrivateKey(),
paymentPReq.getToadd(), paymentPReq.getBtc_amount());
} catch (IOException e) {
e.printStackTrace();
}
if(responsee.getCode().equals("0")){
paymentPReq.setFromadd(keys.getSenderPrivateKey());
System.out.println("From Address:::"+paymentPReq.getFromadd());
```

```
paymentPReq.setTxnId(responsee.getHash());
paymentPReq.setCurrency("hcx");
paymentPReq.setStatus("sending other");
paymentPReq.setMining_fees("0.00001");
paymentPReq.setBtc_amount(paymentPReq.getBtc_amount());
paymentPReq.setDescriptionReceived("HCX Received");
int x =  customerKeyService.insertCustomerKeys(paymentPReq,responsee.getLedger());
System.out.println("customer key :"+ x);
paymentHistService.insertSendPaymentHistory(paymentPReq);
}else{
this.setError("HCX not sent. Please verify receiving address and transaction amount");
}
}
}
public String getError() {
return error;
}
public void setError(String error) {
this.error = error;
}
}
```

CustomerKeyService.java

```
package com.hcx.component.service;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.dao.CustomerKeysDao;
import com.hcx.adapter.dao.to.CustomerKeysTo;
public class CustomerKeyService {
public CustomerKeysDao customerKeysDao;
public CustomerKeysTo customerKeysTo;
public CustomerKeysDao getCustomerKeysDao() {
return customerKeysDao;}
public void setCustomerKeysDao(CustomerKeysDao customerKeysDao) {
this.customerKeysDao = customerKeysDao;
}
public CustomerKeysTo getCustomerKeysTo() {
return customerKeysTo;
}
public void setCustomerKeysTo(CustomerKeysTo customerKeysTo) {
this.customerKeysTo = customerKeysTo;
}
public synchronized CustomerKeysTo getSenderAndReceiverKeys(String fromId, String toId)
{
return this.customerKeysDao.getKeysSenderAndReceiver(fromId,toId);
}
```

CustomerKeysDao.java

```
package com.hcx.adapter.dao;
import java.util.List;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.client.common.util.CommonUtil;

public class CustomerKeysDao extends JdbcDaoSupport{
public synchronized CustomerKeysTo getKeysSenderAndReceiver(String fromId, String toID)
{

String sql= "select c1.HCX_PRIVKEY as sendPrivateKey, c1.HCX_PUBKEY as sendPrivtaeKey,
nvl(c2.HCX_PRIVKEY,'0') as recevPrivateKey, nvl(c2.HCX_PUBKEY, '0') as recevPublicKey "+
"from HCX_BALANCE c1, HCX_BALANCE c2 where  c1.customer_ID!=c2.customer_id and
c1.customer_id='"+fromId+"' and c2.customer_id='"+toID+"'";
System.out.println(sql);

List<CustomerKeysTo> customerr =  (List<CustomerKeysTo>)this.getJdbcTemplate().query(
sql,new RowMapper<CustomerKeysTo>(){
@Override
public CustomerKeysTo mapRow(ResultSet rs,int rno)throws SQLException{
CustomerKeysTo customer=new CustomerKeysTo();
customer.setSenderPrivateKey(rs.getString("sendPrivateKey"));
customer.setSenderPublicKey(rs.getString("sendPrivtaeKey"));
customer.setReceiverPrivateKey(rs.getString("recevPrivateKey"));
customer.setReceiverPublicKey(rs.getString("recevPublicKey"));
return customer;
}
});
return customerr.get(0);
}
}
```

HcNetService.java

```
package com.hcx.component.service;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import org.apache.http.HttpResponse;
```

```
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.hcx.client.common.util.CommonUtil;

public class HcNetService {
public  ClientResponse getCreateAccount() throws IOException  {
ClientResponse respons =null;
CommonUtil commonUtil = new CommonUtil();
String path = commonUtil.getProperty("hcx")+"createAccount";
System.out.println("URL :"+path);
HttpClient client = new DefaultHttpClient();
HttpPost post = new HttpPost(path);
StringEntity input = new StringEntity("{\"amount\":\"20\"}");
input.setContentType("application/json");
post.setEntity(input);
HttpResponse response = client.execute(post);
BufferedReader rd = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
String line = "";
ObjectMapper obj = new ObjectMapper();
while ((line = rd.readLine()) != null) {
respons = obj.readValue(line, ClientResponse.class);
}
return respons;
}
}
```

CommonUtil.java

```
package com.hcx.client.common.util;
import java.io.InputStream;
import java.util.Properties;
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import java.util.Base64;
public class CommonUtil {
private static final Log logger = LogFactory.getLog(CommonUtil.class);
public String getProperty(String name) {
String propertyVal="";
Properties prop = new Properties();
try {
InputStream propertiesFile =
this.getClass().getClassLoader().getResourceAsStream("data.properties");
```

```
prop.load(propertiesFile);
propertyVal = prop.getProperty(name);
if(name.equalsIgnoreCase("USER")||name.equalsIgnoreCase("PASSWORD")){
byte[] base64decodedBytes = Base64.getDecoder().decode(propertyVal);
String strVal=new String(base64decodedBytes, "utf-8");
return strVal;
}else{
return propertyVal;
}
}
(Exception e) {
e.printStackTrace();
}
return propertyVal;
}
}
```

CustomerKeyService.java

```
package com.hcx.component.service;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.dao.CustomerKeysDao;
import com.hcx.adapter.dao.to.CustomerKeysTo;
public class CustomerKeyService {
public CustomerKeysDao customerKeysDao;
public CustomerKeysTo customerKeysTo;

public CustomerKeysDao getCustomerKeysDao() {
return customerKeysDao;}
public void setCustomerKeysDao(CustomerKeysDao customerKeysDao) {
this.customerKeysDao = customerKeysDao;
}
public CustomerKeysTo getCustomerKeysTo() {
return customerKeysTo;
}
public void setCustomerKeysTo(CustomerKeysTo customerKeysTo) {
this.customerKeysTo = customerKeysTo;
}

publicsynchronizedintupdateHcxKeys(String customer_Id, String accPrivKey, String
accPubKey) {
returnthis.customerKeysDao.updateHcxKeysCustomer(customer_Id,accPrivKey,accPubKey);
}
}
```

CustomerKeysDao.java

```
package com.hcx.adapter.dao;
import java.util.List;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.client.common.util.CommonUtil;

public class CustomerKeysDao extends JdbcDaoSupport{

publicsynchronizedintupdateHcxKeysCustomer(String customer_Id,String privKey,String
pubKey) {
String query = "UPDATE HCX_BALANCE  SET HCX_PUBKEY ='"+pubKey+"', HCX_PRIVKEY
='"+privKey+"', "
+ "BUY_HCX=0.0, SELL_HCX=0.0, HCX_BALANCE =0.0  WHERE customer_id
='"+customer_Id+"'";
intcount=this.getJdbcTemplate().update(query);
returncount;
}
}
```

HcNetService.java

```
package com.hcx.component.service;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
import com.fasterxml.jackson.databind.ObjectMapper;
import com.hcx.client.common.util.CommonUtil;

public class HcNetService {


public  ClientResponse betweenTwoUsersHCX(String fromPrivateKey, String toPublicKey,
String strAmount) throws IOException {
```

```
ClientResponse respons=null;
CommonUtil commonUtil = new CommonUtil();
String path = commonUtil.getProperty("hcx")+"fundTransfer";
HttpClient client = newDefaultHttpClient();
HttpPost post = new HttpPost(path);
StringEntity input = new
StringEntity("{\"fromSecretSeed\":\""+fromPrivateKey+"\",\"destination\":\""+toPublicKey+
"\",\"amount\":\""+strAmount+"\"}");
input.setContentType("application/json");
post.setEntity(input);
HttpResponse response = client.execute(post);
BufferedReader rd = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
String line = "";
ObjectMapper obj = new ObjectMapper();
while ((line = rd.readLine()) != null) {
respons = obj.readValue(line, ClientResponse.class);
}
returnrespons;
}
}
```

CustomerKeyService.java

```
package com.hcx.component.service;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.dao.CustomerKeysDao;
import com.hcx.adapter.dao.to.CustomerKeysTo;
public class CustomerKeyService {
public CustomerKeysDao customerKeysDao;
public CustomerKeysTo customerKeysTo;

public CustomerKeysDao getCustomerKeysDao() {
return customerKeysDao;}
public void setCustomerKeysDao(CustomerKeysDao customerKeysDao) {
this.customerKeysDao = customerKeysDao;
}
public CustomerKeysTo getCustomerKeysTo() {
return customerKeysTo;
}
public void setCustomerKeysTo(CustomerKeysTo customerKeysTo) {
this.customerKeysTo = customerKeysTo;
}

publicsynchronizedintinsertCustomerKeys(PaymentProcessingRequestTo pprt, String
strLedger){
customerKeysTo.setCustomerid(pprt.getCustomer_Id());
```

```
customerKeysTo.setFromadd(pprt.getFromadd());
customerKeysTo.setToadd(pprt.getToadd());
customerKeysTo.setBtcAmount(pprt.getBtc_amount());
customerKeysTo.setTxnId(pprt.getTxnId());
customerKeysTo.setStatus(pprt.getStatus());
customerKeysTo.setAction(pprt.getAction());
customerKeysTo.setCurrency(pprt.getCurrency());
intx = customerKeysDao.insertCustomerKeyDao(customerKeysTo,strLedger);
returnx;
}
}
```

CustomerKeysDao.java

```
package com.hcx.adapter.dao;
import java.util.List;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.client.common.util.CommonUtil;

public class CustomerKeysDao extends JdbcDaoSupport{

publicintinsertCustomerKeyDao(CustomerKeysTo customerKeysTo, String ledger_no) {
Long tran_id = (Long)this.getJdbcTemplate().query("select coalesce(max(ID),0)+1 from
CUSTOMERHCXKEYS",new SingleRowExtractor(SingleRowExtractor.LONG));
String query="insert into CUSTOMERHCXKEYS (id,customerid,
fromadd,toadd,hcxamount,txnId,status,action,currency,ledger) values(?,?,?,?,?,?,?,?,?,?)";
intx=this.getJdbcTemplate().update(query,new PreparedStatementSetter() {
@Override
publicvoid setValues(PreparedStatement ps) throws SQLException {
inti = 0;
ps.setObject(++i, tran_id);
ps.setObject(++i,customerKeysTo.getCustomerid()!=null?customerKeysTo.getCustomerid():
null);
ps.setObject(++i, customerKeysTo.getFromadd()!=null?customerKeysTo.getFromadd():null);
ps.setObject(++i, customerKeysTo.getToadd()!=null?customerKeysTo.getToadd():null);
ps.setObject(++i,customerKeysTo.getBtcAmount()!=null?customerKeysTo.getBtcAmount():"
0.0");
ps.setObject(++i, customerKeysTo.getTxnId()!=null? customerKeysTo.getTxnId(): null);
ps.setObject(++i, customerKeysTo.getStatus()!=null?customerKeysTo.getStatus():null);
ps.setObject(++i, customerKeysTo.getAction()!=null?customerKeysTo.getAction():null);
ps.setObject(++i, customerKeysTo.getCurrency()!=null?customerKeysTo.getCurrency():null);
```

```
ps.setObject(++i, ledger_no);
}
});
returnx;
}
}
```

PaymentHistoryService.java

```
package com.hcx.component.service;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import com.hcx.adapter.api.to.CustomerRequestTo;
import com.hcx.adapter.api.to.CustomerResponseTo;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.api.to.PaymentProcessingResponseTo;
import com.hcx.adapter.dao.PaymentHistoryDao;
import com.hcx.adapter.dao.to.PaymentHistoryTo;
public class PaymentHistoryService {

public PaymentHistoryTo paymentHistoryTo;
public PaymentHistoryDao paymentHistoryDao;
public PaymentHistoryTo getPaymentHistoryTo() {
return paymentHistoryTo;
}
public void setPaymentHistoryTo(PaymentHistoryTo paymentHistoryTo) {
this.paymentHistoryTo = paymentHistoryTo;
}
public void setPaymentHistoryDao(PaymentHistoryDao paymentHistoryDao) {
this.paymentHistoryDao = paymentHistoryDao;
}
public PaymentHistoryDao getPaymentHistoryDao() {
return paymentHistoryDao;
}
public synchronized int insertSendPaymentHistory(PaymentProcessingRequestTo rar){

paymentHistoryTo.setCreditHcxAmount(null);
paymentHistoryTo.setDebitHcxAmount(null);
paymentHistoryTo.setHcxTransactionId(null);
if(rar.getCurrency().equals("hcx")){
paymentHistoryTo.setCreditHcxAmount(null);
paymentHistoryTo.setDebitHcxAmount(rar.getSendamount());
paymentHistoryTo.setHcxTransactionId(rar.getTxnId());
}
```

```
paymentHistoryTo.setCustomerId(rar.getCustomer_Id());
paymentHistoryTo.setDescription(rar.getDescription());
paymentHistoryTo.setAction(rar.getAction());
paymentHistoryTo.setTxnCharge(null);
paymentHistoryTo.setChargeCGST(null);
paymentHistoryTo.setChargeSGST(null);
paymentHistoryTo.setMining_fees(rar.getMining_fees());
paymentHistoryTo.setCurrency(rar.getCurrency());
paymentHistoryTo.setStatus("0");
paymentHistoryTo.setOrderID(null);
int x = paymentHistoryDao.insertPaymentHistory(paymentHistoryTo);
return x;
}
}
```

PaymentHistoryDao.java

```
package com.hcx.adapter.dao;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Timestamp;
import java.util.List;
import java.util.Map;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.BatchPreparedStatementSetter;
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.ResultSetExtractor;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.SqlParameterSource;
import org.springframework.jdbc.core.simple.SimpleJdbcCall;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import com.hcx.adapter.dao.to.CustomerLedgerTo;
import com.hcx.adapter.dao.to.PaymentHistoryTo;
import com.hcx.client.common.util.CommonUtil;

public class PaymentHistoryDao extends JdbcDaoSupport {
public synchronized int insertPaymentHistory(PaymentHistoryTo payment) {
Long tran_id = (Long)this.getJdbcTemplate().query("select coalesce(max(TransactionID),0)+1
from payment_history",new SingleRowExtractor(SingleRowExtractor.LONG));
final Timestamp timestamp = (Timestamp)this.getJdbcTemplate().query("select
current_timestamp from dual",new SingleRowExtractor(SingleRowExtractor.TIMESTAMP));
payment.setTransactionId(tran_id.toString());

String sql="insert into payment_history (TransactionID, customer_ID, orderid, "
+ "txnCharge, sgst, cgst, miningFees, "
```

```
+ "transaction_timestamp, description, action, "
+ "hcx_txnID,DEBIT_HCX_AMOUNT,CREDIT_HCX_AMOUNT,currency,
BASECURRENCY,status,networkFees) "
+ "values ( ?, ?, ? ,?, ?, ?, ? ,?, ?, ?,?,?,?,?, ?,?,? )";

int x=this.getJdbcTemplate().update(sql,new PreparedStatementSetter() {

@Override
public void setValues(PreparedStatement ps) throws SQLException {
int i = 0;
ps.setObject(++i,payment.getTransactionId()) ;
ps.setString(++i, payment.getCustomerId());
ps.setString(++i, payment.getOrderID()!=null?payment.getOrderID():null);
ps.setObject(++i, payment.getTxnCharge()!=null ? payment.getTxnCharge() :"0.0");
ps.setObject(++i, payment.getChargeSGST()!=null ? payment.getChargeSGST() :"0.0");
ps.setObject(++i, payment.getChargeCGST()!=null ? payment.getChargeCGST() :"0.0");
ps.setObject(++i, payment.getMining_fees()!=null ? payment.getMining_fees() :"0.0");
ps.setTimestamp(++i, timestamp);
ps.setString(++i, payment.getDescription()!=null? payment.getDescription():"oops");
ps.setString(++i, payment.getAction()!=null?payment.getAction():"pending");
ps.setString(++i, payment.getHcxTransactionId()!=null?payment.getHcxTransactionId():"-");
ps.setString(++i,
payment.getDebitHcxAmount()!=null?payment.getDebitHcxAmount():"0.0");
ps.setString(++i,
payment.getCreditHcxAmount()!=null?payment.getCreditHcxAmount():"0.0");
ps.setString(++i, payment.getCurrency()!=null?payment.getCurrency():null);
ps.setString(++i, payment.getBaseCurrency()!=null?payment.getBaseCurrency():null);
ps.setString(++i, payment.getStatus()!=null? payment.getStatus():"0");
ps.setString(++i, payment.getNetworkFees()!=null? payment.getNetworkFees():"0.0");
}
});
this.callingProcedure(payment, tran_id);
return x;
}
}
```

CustomerKeyService.java

```
package com.hcx.component.service;
import com.hcx.adapter.api.to.PaymentProcessingRequestTo;
import com.hcx.adapter.dao.CustomerKeysDao;
import com.hcx.adapter.dao.to.CustomerKeysTo;
public class CustomerKeyService {
public CustomerKeysDao customerKeysDao;
public CustomerKeysTo customerKeysTo;

public CustomerKeysDao getCustomerKeysDao() {
```

```
return customerKeysDao;}
public void setCustomerKeysDao(CustomerKeysDao customerKeysDao) {
this.customerKeysDao = customerKeysDao;
}
public CustomerKeysTo getCustomerKeysTo() {
return customerKeysTo;
}
public void setCustomerKeysTo(CustomerKeysTo customerKeysTo) {
this.customerKeysTo = customerKeysTo;
}

publicsynchronized CustomerKeysTo getSenderPrivateKey(String customer_Id) {
returnthis.customerKeysDao.getKeysSender(customer_Id);
}
}
```

CustomerKeysDao.java

```
package com.hcx.adapter.dao;
import java.util.List;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import org.springframework.jdbc.core.PreparedStatementSetter;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.jdbc.core.support.JdbcDaoSupport;
import com.hcx.adapter.dao.to.CustomerKeysTo;
import com.hcx.client.common.util.CommonUtil;

public class CustomerKeysDao extends JdbcDaoSupport{

publicsynchronized CustomerKeysTo getKeysSender(String customer_Id) {

String sql= "select HCX_PRIVKEY  as sendPrivateKey from HCX_BALANCE where customer_id
= '"+customer_Id+"'";
List<CustomerKeysTo>customerr =  (List<CustomerKeysTo>)this.getJdbcTemplate().query(
sql,new RowMapper<CustomerKeysTo>(){
@Override
public CustomerKeysTo mapRow(ResultSet rs,intrno)throws SQLException{
CustomerKeysTocustomer=new CustomerKeysTo();
customer.setSenderPrivateKey(rs.getString("sendPrivateKey"));
returncustomer;
}
});
returncustomerr.get(0);
}
}
```

c) Display Pool Account balance to your admin

I) if(strCurrency.equals("hcx")){

```
CommonUtil commonUtil = new CommonUtil();
String key = commonUtil.getProperty("keyvalue");
try {
com.btc.component.service.ClientResponse res=new
com.btc.component.service.StellerService().GetBalance(key);
response.setCredit_btc_amount(res.getBalance());
return response;
} catch (IOException e) {
e.printStackTrace();
}
```

II) public ClientResponse GetBalance(String toPublicKey) throws IOException {

```
ClientResponse respons = null;
CommonUtil commonUtil = new CommonUtil();
String path = commonUtil.getProperty("hcx")+"getBalance";
System.out.println("URL :"+path);
HttpClient client = new DefaultHttpClient();

HttpPost post = new HttpPost(path);
StringEntity input = new StringEntity("{\"accountPublicKey\":\""+toPublicKey+"\"}"); //here
instead of JSON you can also have XML
input.setContentType("application/json");
post.setEntity(input);
HttpResponse response = client.execute(post);
BufferedReader rd = new BufferedReader(new
InputStreamReader(response.getEntity().getContent()));
String line = "";
ObjectMapper obj = new ObjectMapper();
while ((line = rd.readLine()) != null) {
respons = obj.readValue(line, ClientResponse.class);
}
return respons;
}
```

d) HCNet Services

i) createAccount

URL:--->>http://ec2-54-177-50-213.us-west-compute.amazonaws.com:9080/PaybitoStellarUS/rest/PayBitoService/createAccount
JSON_Request:-

```
{
 "amount":"20"
}
```

JSON_Response(if success):-

```
{

"message": "Account created successfully",

 "code": "1",

"destAccountPubKey": "GB……B",

"destAccPrivKey": "S……OO"

}
```

JSON_Response(if failed):-

```
{
   "message": "Account creation failed",

    "code": "0"

}
```

ii) fundTransfer

URL:--->>http://ec2-54-177-50-213.us-west-1.compute.amazonaws.com:9080/PaybitoStellarUS/rest/PayBitoService/fundTransfer

JSON_Request:-

```
{
 "fromSecretSeed":"YOUR_POOLACCOUNT_SECRET_SEED",
 "destination":"YOUR_DESTINATION_ACCOUNT_PUB_KEY",
 "amount":"YOUR_TRANSFER_BALANCE"
}
```

JSON_Response (is success):-

{

"message": "Fund Transfer successful",

"code": "1",

"hash": "3fd6e0c3e269cdd937c20f6e4898e1dd7c18a0e51c3ceb5c5f0eb88c46278d6e",

"ledger": "3593755"

}

JSON_Response (if failed):-

{

"message": "Fund Transfer failed",

"code": "0"

}

iii) Check Balance

URL:--->>http://ec2-54-177-50-213.us-west-1.compute.amazonaws.com:9080/PaybitoStellarUS/rest/PayBitoService/getAccountBalance

JSON_Request:---

{
"accountPublicKey":"YOUR_PUB_KEY"
}

JSON_Response(if success):-

{

"message": "Balance returned successfully",

"statusCode": "1",

"balance": "21.0000000"

}

JSON_Response(if failed):-

{

"message": "Get Balance failed!",

"statusCode": "0",

"balance": "0"

}