Gjavac

发布 *0.1*

WhiteCoin

2022 年 04 月 21 日

Basics

gjavac 用 Java 语言实现的可以在 uvm 上执行的智能合约

备注： This project is under active development.

CHAPTER 1

目录

1.1 智能合约简介

首先，我们所说的智能合约是在区块链 (blockchain) 和加密货币 (cryptocurrencies) 的上下文中.

智能合约 (smart contract) 是:

- 预先写好的代码逻辑（我们使用 glua 进行编写，也支持其他语言的语法）
- 在分布式的存储平台上进行存储和调用（blockchain）
- 可以被运行在同一区块链上的节点执行
- 运行的结果会形成交易进行存储并记录到区块链上

简单点说，智能合约就是一段可执行的代码（它可以被合约编写者赋予各种各样的功能），它经过编译然后被存储在区块链上; 然后根据合约的地址，区块链上的节点可以调用它实现相关的功能.

智能合约有以下几个特点:

- 智能合约是一段可执行的代码，被记录在区块链上，无法被修改
- 当指定的调用代码的交易被发出，区块链上的每个节点（也可能是部分节点）都会运行这段代码并进行必要的校验
- 当前智能合约不能自动执行，但可以由程序外部触发执行
- 智能合约执行所需的输入数据应该依赖区块链的数据，这样才能保证验证结果的一致性

XWC 链支持的智能合约底层采用 lua 的虚拟机，执行效率较高。同时也是图灵完备的语言，可以支持任意的业务逻辑，极大的扩展了区块链的功能。同时我们还提供了其他语言的翻译工具，开发者能够以 Java/Kotlin 和 C# 来进行智能合约的开发。

1.2 gjavac 介绍

gjavac 是一个将 Java 字节码文件转换成 hvm 字节码文件的转换器，结合 Java, Kotlin 等 JVM 平台上语言的编译器将 Java, Kotlin 等源码编译成 Java 字节码，就可以实现将 Java, Kotlin 等 JVM 平台上的编程语言编译到 hvm 字节码, 从而可以用来写智能合约和链上脚本了

1.3 入门教程

1.3.1 开发流程

通常我们在测试链上进行开发调试，当开发测试完成后，再部署到正式链使用。

你需要什么:

- 用于开发 Java 的 IDE 开发环境
- 依赖的 jar 包 gjavac.jar
  - 下载 gjavac.jar
- 编译工具 uvm_ass.exe go_package_gpc.exe
  - 下载 uvm_ass.exe
  - 下载 go_package_gpc.exe
- 正式链账户及一定数量的代币

1.3.2 1. 开发环境

- 安装 Java 开发 IDE
- 新建 Java 项目
- 导入依赖 jar 包
- 以 idea 为例
  - 在项目中新建一个文件夹来存放 jar 包, 一般习惯文件夹名字为 libs
  - 右键你需要添加的 jar 包, 选择 Add as Library
  - 在弹出的 Create Library 窗口上点击 OK 就可以了

1.3.3 2. 第一个合约

1.3.4 3. 编译

1.Java 源码编译成字节码 javac -Djava.ext.dirs={依赖的 gjavac.jar} {源码文件夹} -d {输出目录}

2.Java 字节码编译成.ass 和.json 文件 进入字节码根目录，执行一下命令 java -classpath "{依赖的 gjavac.jar};" gjavac.MainKt {字节码文件列表} "-o" " 输出目录"

3. 生成.out 文件 uvm_ass.exe .ass 文件

4. 生成.gpc 文件 go_package_gpc.exe -package -binary-code=.out 文件 -meta=.json 文件

5. 通过 rpc 形式注册合约上链 curl -X POST -d "{"id": 1,"method": "register_contract", "params": ["kevin", 0.00000001, 50000,"{gpc 文件地址}"]}"

6. 合约部署成功，可以调用合约中方法 1. 调用非上链方法 invoke_contract_offline {调用者名称} {合约地址} {调用的合约方法} "{合约方法的参数列表}" 2. 调用上链方法 invoke_contract {调用者名称} {gas 价格} {gas 最大步数} {调用的合约地址} {调用的合约方法} "{调用的合约方法的参数列表}" .. note:

```
gas 价格，即单步执行花费的 XWC 金额，最少为 0.00001
gas␣
↪最大步数，如果实际执行步数小于该限制，按照实际收取费用，如果大于该限制，调用失败
```

1.3.5 4. 开始用 Java 编写智能合约

通过以上简单的例子我们有了一个直观的感受。智能合约的开发过程非常简单。而开发实际的应用则需要复杂的业务逻辑，也可能要用到更为复杂的数据结构和控制流程。这里可以参考具体的语法参考文档。

参考文档

1.4 gjavac 使用指南

因为 gjavac 不是直接将 Java/Kotlin 源代码编译到 hvm 字节码, 所以需要先安装 Jetbrains IDEA(有免费的社区版) 或者 Java SDK 来将 Java/Kotlin 源码编译到 Java 字节码文件, 然后再使用 gjavac 编译 Java 字节码文件.class 文件转换到 hvm 字节码, 推荐安装 Jetbrains IDEA

可以按照以下步骤配置开发环境:

请按照 Java SDK 8+ 并正确配置环境变量

- 请安装 Jetbrains IDEA Community
- 新建 Java 或 Kotlin 项目, 在项目的 "引用" 中加入 gjavac 的几个.jar 文件。这个项目就是用来写智能合约或者链上脚本的项目, 最终我们就是要把这个项目编译到 hvm 字节码
- 参照 demo 中例子,DemoContract1.java 和 DemoContract.kt 是智能合约的例子。修改新建项目的源代码
- 在同一个 Java/Kotlin 项目下或者新建一个引用刚才项目的新项目。这个项目是用来直接在 Jetbrains IDEA 中调试调用智能合约 Java/Kotlin 代码用的, 将此项目设置为解决方案的启动项目
- 在第 4 步创建的项目中, 可以运行项目对智能合约进行模拟调试运行
- 编译整个项目, 在第 2 步创建的项目的 target/classes 或者 classes 文件夹下 (根据 IDE 和项目管理方式可能不同) 找到此项目产生的各.class 文件

- 执行 gjavac 第 6 步产生的合约相关的各 class 文件 -o result.ass 产生 hvm 汇编文件 result.ass 和合约元信息文件 result.meta.json, 然后使用 hvm_assembler -g result.ass result.meta.json 得到 result.gpc 文件, 这是目标项目的合约.gpc 文件
- 使用产生的.gpc 文件来做注册合约, 调用合约, 注册脚本等后续行为
- 如果是不是要写合约上链, 只是要执行代码, 需要在第 7 步中, 改用 hvm_assembler -c result.ass result_meta.json 来产生"result.out" 文件, 这是 hvm 字节码文件, 然后可以用 hvm 文件路径.out 来直接执行这个字节码文件

1.5 数据类型

- String: 同 Java 的 String 类型,String 是否相等只能用 ==, 暂不支持 equals() 方法,UvmCoreLibs.tostring(Object obj) 转换为 String.
- Long: 同 Java 的 Long 类型, 使用 UvmCoreLibs.tointeger(Object obj) 转换为 Long.
- Boolean: 布尔类型, 同 Java
- UvmMap: UvmMap.create() 创建, 操作方法同 Java 的 HashMap.
- UvmArray: UvmArray.create() 创建, 操作方法同 Java 的 ArrayList, 元素下标从 1 开始.

1.6 示例

1.6.1 Main

```
import gjavac.lib.UvmContract;

import static gjavac.lib.UvmCoreLibs.print;

public class DemoContractEntrypoint {
    public UvmContract main() {
        print("hello java");
        StableTokenContractDemo contract = new StableTokenContractDemo();
        contract.setStorage(new Storage());
        print(contract);
//        contract.init();

        return contract;
    }
}
```

1.6.2 ContractInterface

```
public interface MultiOwnedContractSimpleInterface {
    void on_deposit_contract_token( String arg);
    Object getOn_deposit_contract_token();
}
```

1.6.3 Contract

```
import gjavac.lib.*;

import static gjavac.lib.UvmCoreLibs.*;

@Contract(storage = Storage.class)
public class StableTokenContractDemo extends UvmContract<Storage> {
    @Override
    public void init() {
        print("token contract creating");
        this.getStorage().name = "";
        this.getStorage().symbol = "";
        this.getStorage().supply = 0L;
        this.getStorage().precision = 0L;
        this.getStorage().state = "NOT_INITED";
        this.getStorage().admin = caller_address();
        this.getStorage().minter = "";
        this.getStorage().allowLock = false;
        this.getStorage().fee = 0L;
        this.getStorage().minTransferAmount = 0L;
        this.getStorage().feeReceiveAddress = caller_address();
        print("token contract created");
    }

    @Offline
    public String state(String arg) {
        return this.getStorage().state;
    }

    @Offline
    public String tokenName(String arg) {
        new Utils().checkStateInited(this);
        return this.getStorage().name;
    }

    @Offline
    public Long precision(String arg) {
        new Utils().checkStateInited(this);
        return this.getStorage().precision;
    }

    @Offline
    public String admin(String arg) {
        new Utils().checkStateInited(this);
        return this.getStorage().admin;
    }

    public long totalSupply(String arg) {
        new Utils().checkStateInited(this);
        return this.getStorage().supply;
    }

    @Offline
    public String isAllowLock(String arg) {
        return tostring(this.getStorage().allowLock);
    }
```

(下页继续)

(续上页)

```
    @Offline
    public long supply(String arg) {
        return this.getStorage().supply;
    }

    @Offline
    public String tokenSymbol(String arg) {
        return this.getStorage().symbol;
    }

    @Offline
    public String fee(String arg) {
        return tostring(this.getStorage().fee);
    }

    @Offline
    public String minTransferAmount(String arg) {
        return tostring(this.getStorage().minTransferAmount);
    }

    @Offline
    public String feeReceiveAddress(String arg) {
        return this.getStorage().feeReceiveAddress;
    }

    @Offline
    public String minter(String arg) {
        return this.getStorage().minter;
    }


    private void onDeposit(int amount) {
        error("not support deposit to token");
    }

    public void onDestroy() {
        error("can't destroy token contract");
    }

    public void initToken(String arg) {
        Utils utils = new Utils();
        Storage storage = this.getStorage();
        UvmJsonModule json = (UvmJsonModule) UvmCoreLibs.importModule(UvmJsonModule.
↪class, "json");
        utils.checkAdmin(this);
        pprint("arg:" + arg);
        if (state(arg)!= utils.NOT_INITED()) {
            error("this token contract inited before");
            return;
        }
        UvmArray<String> parsed = utils.parseArgs(arg, 4, "argument format error,␣
↪need format: name,symbol,minter_contract,precision");
        UvmMap<Object> info = UvmMap.create();
        String name = parsed.get(1);
```

(下页继续)

(续上页)

```
        String symbol = parsed.get(2);
        String minter = parsed.get(3);
        long precision = tointeger(parsed.get(4));
        info.set("name", name);
        info.set("symbol", symbol);
        info.set("minter", minter);
        info.set("precision", precision);
        if (utils.isBlank(name)) {
            error("name needed");
            return;
        }
        if (utils.isBlank(symbol)) {
            error("symbol needed");
            return;
        }
        if (utils.isBlank(minter)) {
            error("minter needed");
            return;
        }
        if (!is_valid_contract_address(minter)) {
            error("minter must be contract");
            return;
        }
        if (precision <= 0) {
            error("precision must be positive integer");
            return;
        }
        UvmArray<Long> allowedPrecisions = UvmArray.create();
        allowedPrecisions.add(1L);
        allowedPrecisions.add(10L);
        allowedPrecisions.add(100L);
        allowedPrecisions.add(1000L);
        allowedPrecisions.add(10000L);
        allowedPrecisions.add(100000L);
        allowedPrecisions.add(1000000L);
        allowedPrecisions.add(10000000L);
        allowedPrecisions.add(100000000L);
        if (!utils.arrayContains(allowedPrecisions, precision)) {
            error("precision can only be positive integer in " + json.
↪dumps(allowedPrecisions));
            return;
        }
        storage.setMinter(minter);
        storage.setPrecision(precision);
        storage.setState(utils.COMMON());
        emit("Inited", json.dumps(info));
    }

    public void openAllowLock(String arg) {
        Utils utils = new Utils();
        utils.checkAdmin(this);
        utils.checkState(this);
        if (this.getStorage().getAllowLock()) {
            error("this contract had been opened allowLock before");
            return;
        }
```

(下页继续)

(续上页)

```
        this.getStorage().setAllowLock(true);
        emit("AllowedLock", "");
    }

    public void setFee(String feeStr) {
        Utils utils = new Utils();
        utils.checkAdmin(this);
        utils.checkState(this);
        if (tointeger(feeStr) < 0) {
            error("error fee format");
            return;
        }
        this.getStorage().setFee(tointeger(feeStr));
        emit("FeeChanged", feeStr);
    }

    public void setMinTransferAmount(String minTransferAmountStr) {
        Utils utils = new Utils();
        utils.checkAdmin(this);
        utils.checkState(this);
        if (tointeger(minTransferAmountStr) < 0) {
            error("error minTransferAmount format");
            return;
        }
        this.getStorage().setMinTransferAmount(tointeger(minTransferAmountStr));
        emit("MinTransferAmountChanged", minTransferAmountStr);
    }

    public void setFeeReceiveAddress(String feeReceiveAddress) {
        Utils utils = new Utils();
        utils.checkAdmin(this);
        utils.checkState(this);
        if (!is_valid_address(feeReceiveAddress)) {
            error("invalid address");
            return;
        }
        if (is_valid_contract_address(feeReceiveAddress)) {
            error("can't use contract address");
            return;
        }
        this.getStorage().setFeeReceiveAddress(feeReceiveAddress);
        emit("FeeReceiveAddressChanged", feeReceiveAddress);
    }

    public void transfer(String arg) {
        Utils utils = new Utils();
        utils.checkState(this);
        if ((Storage) this.getStorage() != null) {
            UvmArray parsed = utils.parseAtLeastArgs(arg, 2, "argument format error,␣
↪need format is to_address,integer_amount[,memo]");
            String to = UvmCoreLibs.tostring(parsed.get(1));
            String amountStr = (String) parsed.get(2);
            utils.checkAddress(to);
            UvmSafeMathModule safemathModule = (UvmSafeMathModule) UvmCoreLibs.
↪importModule(UvmSafeMathModule.class, "safemath");
            UvmBigInt bigintAmount = safemathModule.bigint(amountStr);
```

(下页继续)

(续上页)

```
            UvmBigInt bigint0 = safemathModule.bigint(0);
            if (amountStr == null || safemathModule.le(bigintAmount, bigint0)) {
                UvmCoreLibs.error("invalid amount:" + amountStr);
                return;
            }

            String fromAddress = utils.getFromAddress();
            if (fromAddress == to) {
                UvmCoreLibs.error("fromAddress and toAddress is same: " +␣
↪fromAddress);
                return;
            }
            Object temp = UvmCoreLibs.fast_map_get("users", fromAddress);
            if (temp == null) {
                temp = "0";
            }

            UvmBigInt fromBalance = safemathModule.bigint(temp);
            temp = UvmCoreLibs.fast_map_get("users", to);
            if (temp == null) {
                temp = "0";
            }

            UvmBigInt toBalance = safemathModule.bigint(temp);
            if (safemathModule.lt(fromBalance, bigintAmount)) {
                UvmCoreLibs.error("insufficient balance:" + safemathModule.
↪tostring(fromBalance));
            }

            fromBalance = safemathModule.sub(fromBalance, bigintAmount);
            toBalance = safemathModule.add(toBalance, bigintAmount);
            String frombalanceStr = safemathModule.tostring(fromBalance);
            if (frombalanceStr == "0") {
                UvmCoreLibs.fast_map_set("users", fromAddress, (Object) null);
            } else {
                UvmCoreLibs.fast_map_set("users", fromAddress, frombalanceStr);
            }

            UvmCoreLibs.fast_map_set("users", to, safemathModule.tostring(toBalance));
            if (UvmCoreLibs.is_valid_contract_address(to)) {
                MultiOwnedContractSimpleInterface multiOwnedContract =␣
↪(MultiOwnedContractSimpleInterface) UvmCoreLibs.
↪importContractFromAddress(MultiOwnedContractSimpleInterface.class, to);
                if (multiOwnedContract != null && multiOwnedContract.getOn_deposit_
↪contract_token() != null) {
                    multiOwnedContract.on_deposit_contract_token(amountStr);
                }
            }

            UvmMap eventArg = UvmMap.create();
            eventArg.set("from", fromAddress);
            eventArg.set("to", to);
            eventArg.set("amount", amountStr);
            String eventArgStr = UvmCoreLibs.tojsonstring(eventArg);
            UvmCoreLibs.emit("Transfer", eventArgStr);
        }
```

(下页继续)

(续上页)

```
    }

    public void transferFrom(String arg) {
        Utils utils = new Utils();
        utils.checkState(this);
        if ((Storage) this.getStorage() != null) {
            UvmArray parsed = utils.parseAtLeastArgs(arg, 3, "argument format error,␣
↪need format is fromAddress,toAddress,amount(with precision)");
            String fromAddress = UvmCoreLibs.tostring(parsed.get(1));
            String toAddress = UvmCoreLibs.tostring(parsed.get(2));
            String amountStr = UvmCoreLibs.tostring(parsed.get(3));
            utils.checkAddress(fromAddress);
            utils.checkAddress(toAddress);
            if (fromAddress == toAddress) {
                UvmCoreLibs.error("fromAddress and toAddress is same: " +␣
↪fromAddress);
                return;
            }
            UvmSafeMathModule safemathModule = (UvmSafeMathModule) UvmCoreLibs.
↪importModule(UvmSafeMathModule.class, "safemath");
            UvmBigInt bigintAmount = safemathModule.bigint(amountStr);
            UvmBigInt bigint0 = safemathModule.bigint(0);
            if (amountStr == null || safemathModule.le(bigintAmount, bigint0)) {
                UvmCoreLibs.error("invalid amount:" + amountStr);
            }

            Object temp = UvmCoreLibs.fast_map_get("users", fromAddress);
            if (temp == null) {
                temp = "0";
            }

            UvmBigInt bigintFromBalance = safemathModule.bigint(temp);
            Object temp2 = UvmCoreLibs.fast_map_get("users", toAddress);
            if (temp2 == null) {
                temp2 = "0";
            }

            UvmBigInt bigintToBalance = safemathModule.bigint(temp2);
            if (safemathModule.lt(bigintFromBalance, bigintAmount)) {
                UvmCoreLibs.error("insufficient balance :" + safemathModule.
↪tostring(bigintFromBalance));
            }

            Object allowedDataStr = UvmCoreLibs.fast_map_get("allowed", fromAddress);
            if (allowedDataStr == null) {
                UvmCoreLibs.error("not enough approved amount to withdraw");
            } else {
                UvmJsonModule jsonModule = (UvmJsonModule) UvmCoreLibs.
↪importModule(UvmJsonModule.class, "json");
                UvmMap allowedData = (UvmMap) UvmCoreLibs.totable(jsonModule.
↪loads(UvmCoreLibs.tostring(allowedDataStr)));
                String contractCaller = utils.getFromAddress();
                if (allowedData == null) {
                    UvmCoreLibs.error("not enough approved amount to withdraw");
                } else {
                    String approvedAmountStr = (String) allowedData.
↪get(contractCaller);
```

(下页继续)

(续上页)

```
                    if (approvedAmountStr == null) {
                        UvmCoreLibs.error("no approved amount to withdraw");
                    }

                    UvmBigInt bigintApprovedAmount = safemathModule.
↪bigint(approvedAmountStr);
                    if (bigintApprovedAmount != null && !safemathModule.
↪gt(bigintAmount, bigintApprovedAmount)) {
                        bigintFromBalance = safemathModule.sub(bigintFromBalance,␣
↪bigintAmount);
                        String bigintFromBalanceStr = safemathModule.
↪tostring(bigintFromBalance);
                        if (bigintFromBalanceStr == "0") {
                            bigintFromBalance = null;
                        }
                        bigintToBalance = safemathModule.add(bigintToBalance,␣
↪bigintAmount);
                        String bigintToBalanceStr = safemathModule.
↪tostring(bigintToBalance);
                        if (bigintToBalanceStr == "0") {
                            bigintToBalanceStr = null;
                        }

                        bigintApprovedAmount = safemathModule.
↪sub(bigintApprovedAmount, bigintAmount);
                        UvmCoreLibs.fast_map_set("users", fromAddress,␣
↪bigintFromBalanceStr);
                        UvmCoreLibs.fast_map_set("users", toAddress,␣
↪bigintToBalanceStr);
                        if (safemathModule.tostring(bigintApprovedAmount) == "0") {
                            allowedData.set(contractCaller, null);
                        } else {
                            allowedData.set(contractCaller, safemathModule.
↪tostring(bigintApprovedAmount));
                        }

                        allowedDataStr = UvmCoreLibs.tojsonstring(allowedData);
                        UvmCoreLibs.fast_map_set("allowed", fromAddress,␣
↪allowedDataStr);
                        if (UvmCoreLibs.is_valid_contract_address(toAddress)) {
                            MultiOwnedContractSimpleInterface multiOwnedContract =␣
↪(MultiOwnedContractSimpleInterface) UvmCoreLibs.
↪importContractFromAddress(MultiOwnedContractSimpleInterface.class, toAddress);
                            if (multiOwnedContract != null && multiOwnedContract.
↪getOn_deposit_contract_token() != null) {
                                multiOwnedContract.on_deposit_contract_
↪token(amountStr);
                            }
                        }

                        UvmMap eventArg = UvmMap.create();
                        eventArg.set("from", fromAddress);
                        eventArg.set("to", toAddress);
                        eventArg.set("amount", amountStr);
                        String eventArgStr = UvmCoreLibs.tojsonstring(eventArg);
                        UvmCoreLibs.emit("Transfer", eventArgStr);
```

(下页继续)

(续上页)

```
                    } else {
                        UvmCoreLibs.error("not enough approved amount to withdraw");
                    }
                }
            }
        }
    }

    public void approve(String arg) {
        Utils utils = new Utils();
        utils.checkState(this);
        if ((Storage) this.getStorage() != null) {
            UvmArray parsed = utils.parseAtLeastArgs(arg, 2, "argument format error,␣
↪need format is spenderAddress,amount(with precision)");
            String spender = UvmCoreLibs.tostring(parsed.get(1));
            utils.checkAddress(spender);
            String amountStr = UvmCoreLibs.tostring(parsed.get(2));
            UvmSafeMathModule safemathModule = (UvmSafeMathModule) UvmCoreLibs.
↪importModule(UvmSafeMathModule.class, "safemath");
            UvmBigInt bigintAmount = safemathModule.bigint(amountStr);
            UvmBigInt bigint0 = safemathModule.bigint(0);
            if (amountStr == null || safemathModule.lt(bigintAmount, bigint0)) {
                UvmCoreLibs.error("amount must be non-negative integer");
            }

            String contractCaller = utils.getFromAddress();
            UvmJsonModule jsonModule = (UvmJsonModule) UvmCoreLibs.
↪importModule(UvmJsonModule.class, "json");
            UvmMap allowedDataTable = (UvmMap) null;
            Object allowedDataStr = UvmCoreLibs.fast_map_get("allowed",␣
↪contractCaller);
            if (allowedDataStr == null) {
                allowedDataTable = UvmMap.create();
            } else {
                allowedDataTable = (UvmMap) UvmCoreLibs.totable(jsonModule.
↪loads(UvmCoreLibs.tostring(allowedDataStr)));
                if (allowedDataTable == null) {
                    UvmCoreLibs.error("allowed storage data error");
                    return;
                }
            }

            if (safemathModule.eq(bigintAmount, bigint0)) {
                allowedDataTable.set(spender, null);
            } else {
                allowedDataTable.set(spender, amountStr);
            }

            UvmCoreLibs.fast_map_set("allowed", contractCaller, UvmCoreLibs.
↪tojsonstring(allowedDataTable));
            UvmMap eventArg = UvmMap.create();
            eventArg.set("from", contractCaller);
            eventArg.set("spender", spender);
            eventArg.set("amount", amountStr);
            String eventArgStr = UvmCoreLibs.tojsonstring(eventArg);
            UvmCoreLibs.emit("Approved", eventArgStr);
```

(下页继续)

(续上页)

```
        }
    }

    public void pause(String arg) {
        Utils utils = new Utils();
        Storage var10000 = (Storage) this.getStorage();
        if (var10000 != null) {
            Storage storage = var10000;
            String state = storage.getState();
            if (state == utils.STOPPED()) {
                UvmCoreLibs.error("this contract stopped now, can't pause");
            } else if (state == utils.PAUSED()) {
                UvmCoreLibs.error("this contract paused now, can't pause");
            } else {
                utils.checkAdmin(this);
                storage.setState(utils.PAUSED());
                UvmCoreLibs.emit("Paused", "");
            }
        }
    }


    public void resume(String arg) {
        Utils utils = new Utils();
        Storage var10000 = (Storage) this.getStorage();
        if (var10000 != null) {
            Storage storage = var10000;
            String state = storage.getState();
            if (state != utils.PAUSED()) {
                UvmCoreLibs.error("this contract not paused now, can't resume");
            } else {
                utils.checkAdmin(this);
                storage.setState(utils.COMMON());
                UvmCoreLibs.emit("Resumed", "");
            }
        }
    }

    public void stop(String arg) {
        Utils utils = new Utils();
        Storage var10000 = (Storage) this.getStorage();
        if (var10000 != null) {
            Storage storage = var10000;
            String state = storage.getState();
            if (state == utils.STOPPED()) {
                UvmCoreLibs.error("this contract stopped now, can't stop");
            } else if (state == utils.PAUSED()) {
                UvmCoreLibs.error("this contract paused now, can't stop");
            } else {
                utils.checkAdmin(this);
                storage.setState(utils.STOPPED());
                UvmCoreLibs.emit("Stopped", "");
            }
        }
    }
```

(下页继续)

(续上页)

```
    public void lock(String arg) {
        Utils utils = new Utils();
        utils.checkState(this);
        Storage var10000 = (Storage) this.getStorage();
        if (var10000 != null) {
            Storage storage = var10000;
            if (!storage.getAllowLock()) {
                UvmCoreLibs.error("this token contract not allow lock balance");
            } else {
                UvmArray parsed = utils.parseAtLeastArgs(arg, 2, "arg format error,␣
↪need format is integer_amount,unlockBlockNumber");
                String toLockAmount = (String) parsed.get(1);
                long unlockBlockNumber = UvmCoreLibs.tointeger(parsed.get(2));
                UvmSafeMathModule safemathModule = (UvmSafeMathModule) UvmCoreLibs.
↪importModule(UvmSafeMathModule.class, "safemath");
                UvmBigInt bigintToLockAmount = safemathModule.bigint(toLockAmount);
                UvmBigInt bigint0 = safemathModule.bigint(0L);
                if (toLockAmount != null && !safemathModule.le(bigintToLockAmount,␣
↪bigint0)) {
                    if (unlockBlockNumber < UvmCoreLibs.get_header_block_num()) {
                        UvmCoreLibs.error("to unlock block number can't be earlier␣
↪than current block number " + UvmCoreLibs.tostring(UvmCoreLibs.get_header_block_
↪num()));
                    } else {
                        String fromAddress = utils.getFromAddress();
                        if (fromAddress != UvmCoreLibs.caller_address()) {
                            UvmCoreLibs.error("only common user account can lock␣
↪balance");
                        } else {
                            Object temp = UvmCoreLibs.fast_map_get("users",␣
↪fromAddress);
                            if (temp == null) {
                                UvmCoreLibs.error("your balance is 0");
                            } else {
                                UvmBigInt bigintFromBalance = safemathModule.
↪bigint(temp);
                                if (safemathModule.gt(bigintToLockAmount,␣
↪bigintFromBalance)) {
                                    UvmCoreLibs.error("you have not enough balance to␣
↪lock");
                                } else {
                                    Object lockedAmount = UvmCoreLibs.fast_map_get(
↪"lockedAmounts", fromAddress);
                                    if (lockedAmount == null) {
                                        UvmCoreLibs.fast_map_set("lockedAmounts",␣
↪fromAddress, UvmCoreLibs.tostring(toLockAmount) + "," + UvmCoreLibs.
↪tostring(unlockBlockNumber));
                                        bigintFromBalance = safemathModule.
↪sub(bigintFromBalance, bigintToLockAmount);
                                        UvmCoreLibs.fast_map_set("users", fromAddress,
↪ safemathModule.tostring(bigintFromBalance));
                                        UvmCoreLibs.emit("Locked", UvmCoreLibs.
↪tostring(toLockAmount));
                                    } else {
                                        UvmCoreLibs.error("you have locked balance␣
↪now, before lock again, you need unlock them or use other address to lock");
```

(下页继续)

(续上页)

```
                                    }
                                }
                            }
                        }
                    }
                } else {
                    UvmCoreLibs.error("to unlock amount must be positive integer");
                }
            }
        }
    }


    public void unlock(String arg) {
        Utils utils = new Utils();
        String fromAddress = utils.getFromAddress();
        forceUnlock(fromAddress);
    }

    public void forceUnlock(String unlockAddress) {
        Utils utils = new Utils();
        utils.checkState(this);

        if (this.getStorage().getAllowLock() == false) {
            UvmCoreLibs.error("this token contract not allow lock balance");
        } else {
            Object lockedStr = UvmCoreLibs.fast_map_get("lockedAmounts",␣
↪unlockAddress);
            if (lockedStr == null) {
                UvmCoreLibs.error("you have not locked balance");
            } else {
                UvmArray lockedInfoParsed = utils.parseAtLeastArgs(UvmCoreLibs.
↪tostring(lockedStr), 2, "locked amount info format error");
                String lockedAmountStr = UvmCoreLibs.tostring(lockedInfoParsed.
↪get(1));
                long canUnlockBlockNumber = UvmCoreLibs.tointeger(lockedInfoParsed.
↪get(2));
                if (UvmCoreLibs.get_header_block_num() < canUnlockBlockNumber) {
                    UvmCoreLibs.error("your locked balance only can be unlock after␣
↪block #" + UvmCoreLibs.tostring(canUnlockBlockNumber));
                    return;
                }
                UvmCoreLibs.fast_map_set("lockedAmounts", unlockAddress, (Object)␣
↪null);
                Object temp = UvmCoreLibs.fast_map_get("users", unlockAddress);
                if (temp == null) {
                    temp = "0";
                }

                UvmSafeMathModule safemathModule = (UvmSafeMathModule) UvmCoreLibs.
↪importModule(UvmSafeMathModule.class, "safemath");
                UvmBigInt bigintFromBalance = safemathModule.bigint(temp);
                UvmBigInt bigintLockedAmount = safemathModule.bigint(UvmCoreLibs.
↪tostring(lockedAmountStr));
                bigintFromBalance = safemathModule.add(bigintFromBalance,␣
↪bigintLockedAmount);
```

(下页继续)

(续上页)

```
                UvmCoreLibs.fast_map_set("users", unlockAddress, safemathModule.
↪tostring(bigintFromBalance));
                String tempevent = unlockAddress + "," + UvmCoreLibs.
↪tostring(lockedStr);
                UvmCoreLibs.emit("Unlocked", tempevent);
            }
        }
    }


    public void mint(String arg) {
        Utils utils = new Utils();
        UvmJsonModule json = (UvmJsonModule) UvmCoreLibs.importModule(UvmJsonModule.
↪class, "json");
        utils.checkState(this);
        utils.checkMinter(this);
        UvmArray<String> parsed = utils.parseArgs(arg, 2, "argument format error,␣
↪need format: to_address,token_amount");
        String toAddress = parsed.get(1);
        String amountStr = parsed.get(2);
        long amount = utils.checkInteger(amountStr);
        if (!is_valid_address(toAddress)) {
            error("to_address is not valid address");
            return;
        }
        if (amount <= 0) {
            error("arg token_amount must > 0");
            return;
        }

        long originSupply = this.getStorage().getSupply();
        long newSupply = originSupply + amount;
        if (newSupply <= originSupply) {
            error("supply over flow");
            return;
        }

        long userOldBalance = tointeger(fast_map_get("users", toAddress));
        fast_map_set("users", toAddress, userOldBalance + amount);
        UvmMap eventArg = UvmMap.create();
        eventArg.set("address", toAddress);
        eventArg.set("amount", amount);
        String eventArgStr = UvmCoreLibs.tojsonstring(eventArg);
        emit("Mint", json.dumps(eventArgStr));
    }

    public void destoryAndTrans(String arg) {
        Utils utils = new Utils();
        utils.checkState(this);
        utils.checkMinter(this);
        UvmArray<String> parsed = utils.parseArgs(arg, 4, "argument format error,␣
↪need format: from_address,destory_amount,trans_to_address,trans_amount");
        String fromAddress = parsed.get(1);
        long destoryAmount = utils.checkInteger(parsed.get(2));
        if (destoryAmount < 0) {
            error("arg destory_amount must >= 0");
```

(下页继续)

(续上页)

```
            return;
        }
        String transToAddress = parsed.get(3);
        long transAmount = utils.checkInteger(parsed.get(4));
        if (transAmount < 0) {
            error("arg trans_amount must >= 0");
            return;
        }
        if (destoryAmount == 0 && transAmount == 0) {
            error("destory_amount and trans_amount is 0");
            return;
        }
        long originSupple = this.getStorage().getSupply();
        if (originSupple < destoryAmount) {
            error("supply minus error");
            return;
        }
        this.getStorage().setSupply(originSupple - destoryAmount);
        long fromOldBalance = tointeger(fast_map_get("users", fromAddress));
        long subFromAmount = destoryAmount + transAmount;
        if (fromOldBalance < subFromAmount) {
            error("not enough balance to destory and trans , now balance:" +␣
↪tostring(fromOldBalance) + " need amount:" + tostring(subFromAmount));
            return;
        }

        if (fromOldBalance == subFromAmount) {
            fast_map_set("users", fromAddress, null);
        } else {
            fast_map_set("users", fromAddress, fromOldBalance - subFromAmount);
        }

        if (transAmount > 0) {
            if (!is_valid_address(transToAddress)) {
                error("trans_to_address is not valid address");
                return;
            }
            long toOldBalance = tointeger(fast_map_get("users", transToAddress));
            fast_map_set("users", transToAddress, toOldBalance + transAmount);
        }

        UvmJsonModule json = (UvmJsonModule) UvmCoreLibs.importModule(UvmJsonModule.
↪class, "json");
        UvmMap eventArg = UvmMap.create();
        eventArg.set("from_address", fromAddress);
        eventArg.set("destory_amount", destoryAmount);
        eventArg.set("trans_to_address", transToAddress);
        eventArg.set("trans_amount", transAmount);
        String eventArgStr = UvmCoreLibs.tojsonstring(eventArg);
        emit("DestoryAndTrans", json.dumps(eventArgStr));
    }

    @Offline
    public String lockedBalanceOf(String owner) {
        Object resultStr = fast_map_get("lockedAmounts", owner);
        if (resultStr == null) {
```

(下页继续)

(续上页)

```
            return "0,0";
        }
        return String.valueOf(resultStr);
    }

    @Offline
    public String balanceOf(String owner) {
        Utils utils = new Utils();
        utils.checkStateInited(this);
        utils.checkAddress(owner);
        String amountStr = utils.getBalanceOfUser(this, owner);
        return amountStr;
    }

    @Offline
    public String approvedBalanceFrom(String arg) {
        Utils utils = new Utils();
        if ((Storage) this.getStorage() != null) {
            UvmArray parsed = utils.parseAtLeastArgs(arg, 2, "argument format error,␣
↪need format is spenderAddress,authorizerAddress");
            String spender = UvmCoreLibs.tostring(parsed.get(1));
            String authorizer = UvmCoreLibs.tostring(parsed.get(2));
            utils.checkAddress(spender);
            utils.checkAddress(authorizer);
            Object allowedDataStr = UvmCoreLibs.fast_map_get("allowed", authorizer);
            if (allowedDataStr == null) {
                return "0";
            } else {
                UvmJsonModule jsonModule = (UvmJsonModule) UvmCoreLibs.
↪importModule(UvmJsonModule.class, "json");
                UvmMap allowedDataTable = (UvmMap) UvmCoreLibs.totable(jsonModule.
↪loads(UvmCoreLibs.tostring(allowedDataStr)));
                if (allowedDataTable == null) {
                    return "0";
                } else {
                    String allowedAmount = (String) allowedDataTable.get(spender);
                    return allowedAmount == null ? "0" : allowedAmount;
                }
            }
        } else {
            return "";
        }
    }

    @Offline
    public String allApprovedFromUser(String arg) {
        Utils utils = new Utils();
        if ((Storage) this.getStorage() != null) {
            utils.checkAddress(arg);
            Object allowedDataStr = UvmCoreLibs.fast_map_get("allowed", "authorizer");
            if (allowedDataStr == null) {
                return "{}";
            } else {
                return UvmCoreLibs.tostring(allowedDataStr);
            }
        } else {
```

(下页继续)

(续上页)

```
            return "";
        }
    }


}
```

1.6.4 Storage

```
public class Storage {
    public String name;
    public String symbol;
    public Long supply;
    public Long precision;
    public String state;
    public boolean allowLock;
    public Long fee;
    /* 每次最低转账金额 */
    public Long minTransferAmount;
    /* 手续费接收地址 */
    public String feeReceiveAddress;
    /* admin user address */
    public String admin;
    public String minter;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getSymbol() {
        return symbol;
    }

    public void setSymbol(String symbol) {
        this.symbol = symbol;
    }

    public Long getSupply() {
        return supply;
    }

    public void setSupply(Long supply) {
        this.supply = supply;
    }

    public Long getPrecision() {
        return precision;
    }

    public void setPrecision(Long precision) {
```

(下页继续)

(续上页)

```
        this.precision = precision;
    }

    public String getState() {
        return state;
    }

    public void setState(String state) {
        this.state = state;
    }

    public boolean getAllowLock() {
        return allowLock;
    }

    public void setAllowLock(boolean allowLock) {
        this.allowLock = allowLock;
    }

    public Long getFee() {
        return fee;
    }

    public void setFee(Long fee) {
        this.fee = fee;
    }

    public Long getMinTransferAmount() {
        return minTransferAmount;
    }

    public void setMinTransferAmount(Long minTransferAmount) {
        this.minTransferAmount = minTransferAmount;
    }

    public String getFeeReceiveAddress() {
        return feeReceiveAddress;
    }

    public void setFeeReceiveAddress(String feeReceiveAddress) {
        this.feeReceiveAddress = feeReceiveAddress;
    }

    public String getAdmin() {
        return admin;
    }

    public void setAdmin(String admin) {
        this.admin = admin;
    }

    public String getMinter() {
        return minter;
    }

    public void setMinter(String minter) {
```

(下页继续)

(续上页)

```
        this.minter = minter;
    }
}
```

1.6.5 Utils

```
import gjavac.lib.*;
import kotlin.Pair;


import static gjavac.lib.UvmCoreLibs.*;


@Component
public class Utils {
    public String NOT_INITED() {
        return "NOT_INITED";
    }


    public String COMMON() {
        return "COMMON";
    }


    public String PAUSED() {
        return "PAUSED";
    }


    public String STOPPED() {
        return "STOPPED";
    }

    public final long checkInteger(String numstr) {
        if (isBlank(numstr)) {
            error("integer format error of " + numstr);
            return 0;
        }
        return tointeger(numstr);
    }

    public final String getFromAddress() {
        String fromAddress;
        final String prevContractId = get_prev_call_frame_contract_address();
        if (isNotBlank(prevContractId) && is_valid_address(prevContractId)) {
            fromAddress = prevContractId;
        } else {
            fromAddress = caller_address();
        }
        return fromAddress;
    }

    public final void checkAdmin(StableTokenContractDemo self) {
```

(下页继续)

(续上页)

```
        String fromAddress = getFromAddress();
        if (self.getStorage().admin != fromAddress) {
            error("you are not admin, can't call this function");
        }
    }

    public final void checkMinter(StableTokenContractDemo self) {
        String fromAddress = getFromAddress();
        if (self.getStorage().minter != fromAddress) {
            error("you are not minter, can't call this function");
        }
    }

    public final UvmArray<String> parseArgs(String arg, int count, String errorMsg) {
        if (isBlank(arg)) {
            error(errorMsg);
            return UvmArray.create();
        }
        UvmStringModule stringModule = importModule(UvmStringModule.class, "string");
        UvmArray<String> parsed = stringModule.split(arg, ",");
        if (parsed != null && parsed.size() == count) {
            return parsed;
        } else {
            error(errorMsg);
            return UvmArray.create();
        }
    }

    public final UvmArray<String> parseAtLeastArgs(String arg, int count, String␣
↪errorMsg) {
        if (isBlank(arg)) {
            error(errorMsg);
            return UvmArray.create();
        }

        UvmStringModule stringModule = importModule(UvmStringModule.class, "string");
        UvmArray<String> parsed = stringModule.split(arg, ",");
        if (parsed != null && parsed.size() >= count) {
            return parsed;
        } else {
            error(errorMsg);
            return UvmArray.create();
        }

    }

    public final boolean arrayContains(UvmArray col, Object item) {
        if (col != null && item != null) {
            ArrayIterator colTter = col.ipairs();
            for (Pair colKeyValuePari = (Pair) colTter.invoke(col, 0);
                 colKeyValuePari.getFirst() != null;
                 colKeyValuePari = (Pair) colTter.invoke(col, colKeyValuePari.
↪getFirst())) {
                if (colKeyValuePari != null && colKeyValuePari.getSecond() == item) {
                    return true;
                }
```

(下页继续)

(续上页)

```
            }
            return false;
        } else {
            return false;
        }
    }

    public final void checkState(StableTokenContractDemo self) {
        String state = self.getStorage().state;
        if (state == NOT_INITED())
            error("contract token not inited");

        if (state == PAUSED())
            error("contract paused");

        if (state == STOPPED())
            error("contract stopped");
    }

    public final void checkStateInited(StableTokenContractDemo self) {
        if (self.getStorage().state == NOT_INITED())
            error("contract token not inited");
    }

    public final boolean checkAddress(String addr) {
        boolean result = is_valid_address(addr);
        if (!result) {
            error("address format error");
        }
        return result;
    }

    public final String getBalanceOfUser(StableTokenContractDemo self, String addr) {
        Object balance = fast_map_get("users", addr);
        if (balance == null) {
            return "0";
        }
        return tostring(balance);
    }

    public final boolean isBlank(String str) {
        return str == null || str.length() == 0;
    }

    public final boolean isNotBlank(String str) {
        return !isBlank(str);
    }
}
```

1.7 合约 API 函数

合约 API 是在 jar 的 UvmCoreLibs 包中的静态方法, 主要的 API:

- 使用全局函数 transfer_from_contract_to_address 可以从当前合约 (这个函数调用代码所在的合约) 转账一定数额的某种资产给某个地址, 第一个参数是目标地址 (字符串), 第二个参数是资产名称 (比如 HSR) 第三个参数是转账数量的 10 万倍 (int64 类型), 要求是正数

  返回值 0 转账成功 -1 未知系统异常 -2 Asset_symbol 异常 -3 合约地址非法 -4 目标地址非法 -5 账户余额不足支付转账金额 -6 转账金额为负数

- 使用全局函数 get_contract_balance_amount 可以获取某个合约带精度的余额 (精度为 100000), 第一个参数是合约地址 (支持查询其他合约的余额), 第二个参数是资产名称 (比如 HSR), 返回带精度的合约余额 (int64 类型), 如果出现错误或者合约不存在返回负数

  返回值非负数合约账户余额 -1 资产 id 异常 -2 合约地址异常

- 使用全局函数 get_chain_now 可以获取链上的当前时间, 没有参数.

  返回值正数时间戳整数 0 系统异常

- 使用全局函数 get_chain_random 可以获取链上的一个伪随机数字, 但是同一个此链上的 operation 操作, 不同节点不同时间执行返回结果都一样 (实际是取操作发生的块上 prev_secret_hash 和本交易结合后的哈希)

  返回值随机结果

- 使用全局函数 get_header_block_num, 可以获取上一个块的块号

  返回值当前链最新块的序号

- 使用全局函数 get_current_contract_address 可以获取这个函数调用出现位置的合约地址, 没有参数

- 全局变量 caller 存储着调用合约的用户的公钥, 全局变量 caller_address 存储着调用合约的用户的账户地址

- 在转账到合约发生的时候, 如果合约中定义了 on_deposit_asset(参数是" 资产标识, 转账金额") 这个 API, 那么在转账发生后会调用这个 API, 并且保证转账和触发此 API 是原子性的, 如果中途出现错误, 整体回滚, 转账失败。

- 使用语句 emit EventName(arg: string) 可以抛出事件, 这里 emit 是关键字,EventName 根据需要写入事件名称, 由区块链记录下来, 其他节点同步到 emit 触发的 event 时可以调用本地设置的回调

- 使用全局函数 is_valid_address(arg: string) 可以检查一个地址字符串是否是合法的本区块链地址

- 使用全局函数 is_valid_contract_address(arg: string) 可以检查一个地址字符串是否是合法的合约地址

- 使用全局函数 get_transaction_fee() 可以获取一笔交易的手续费

  返回值正整数结果值 -1 手续费资产 id 异常 -2 系统异常

- 使用全局函数 get_transaction_id(): string 可以获取本次交易的交易 id

- 使用全局函数 transfer_from_contract_to_public_account(to_account_name: string, asset_type: string, amount: int) 可以从当前合约中转账到链上的账户名称, 返回是否转账的状态

  返回值 0 转账成功 -1 未知系统异常 -2 Asset_symbol 异常 -3 合约地址非法 -4 目标地址非法 -5 账户余额不足支付转账金额 -6 转账金额为负数 -7 不存在指定账户名

- import_contract: (string) => table 引用合约, 参数是合约的名称字符串, 返回合约对应的 table

- import_contract_from_address: (string) => table 根据合约地址引用合约, 返回合约对应的 table

- get_prev_call_frame_contract_address: () => string 获取合约调用栈的上一级合约地址 (如果上一级合约调用栈不是合约, 则返回 null)

- get_prev_call_frame_api_name: () => string 获取合约调用栈的上一级合约 API 名称 (如果上一级合约调用栈不是合约, 则返回 null)
- get_contract_call_frame_stack_size: () => int 获取合约调用栈深度
- wait_for_future_random: (int) => int 根据参数的块高度获取根据这个块数据得到的伪随机数, 如果这个块高度还没有达到, 则返回 0
- get_system_asset_symbol: () => string 获取系统基础资产的资产符号
- get_system_asset_precision: () => int 获取系统基础资产的精度, 这个值一般是 10 的若干次方