成都链安
BEOSIN

智能合约安全审计报告

审计编号：201902231155

审计合约名称：

合约编号	合约名称	文件 MD5
1	ImprovedStoremanGroupAdmin.sol	C47BB2B9FA7BFB3FC3C95A0EACD6D430
2	StoremanGroup.sol	67983FB66ECA257C9BD667426D8D346E
3	StoremanLottery.sol	C65AB5A9D6BFED862408B61E1FD92B86

合约审计开始日期：2019.02.13

合约审计完成日期：2019.02.23

审计结果：通过（良）

审计团队：成都链安科技有限公司

审计类型及结果：

序号	审计类型	审计子项	审计结果
1	代码规范审计	编译器版本安全审计	通过
		弃用项审计	**不通过**
		冗余代码审计	通过
		require/assert 使用审计	通过
		gas 消耗审计	通过
2	通用漏洞审计	整型溢出审计	通过
		重入攻击审计	通过
		伪随机数生成审计	通过
		交易顺序依赖审计	通过
		拒绝服务攻击审计	通过
		函数调用权限审计	通过
		call/delegatecall 安全审计	通过
		返回值安全审计	通过
		tx.origin 使用安全审计	通过
		重放攻击审计	通过
		变量覆盖审计	通过

3	业务审计	业务逻辑审计	通过
		业务实现审计	通过

备注：审计意见及建议请见代码注释。

免责声明：本次审计仅针对审计类型及结果表中给定的审计类型范围进行审计，其他未知安全漏洞不在本次审计责任范围之内。成都链安科技仅根据本报告出具前已经存在或发生的攻击或漏洞出具本报告，并就此承担相应责任。对于出具以后存在或发生的新的攻击或漏洞，成都链安科技无法判断其对智能合约安全状况可能的影响，亦不对此承担责任。本报告所作的安全审计分析及其他内容，仅基于合约提供者截至本报告出具时向成都链安科技提供的文件和资料，文件和资料不应存在缺失、被篡改、删减或隐瞒的情形；如提供的文件和资料存在信息缺失、被篡改、删减、隐瞒或反映的情况与实际情况不符等情况，成都链安科技对由此而导致的损失和不利影响不承担任何责任。

审计结果说明：

本公司采用形式化验证、静态分析、动态分析、典型案例测试和人工审核的方式对万维链项目部分合约(ImprovedStoremanGroupAdmin.sol、StoremanGroup.sol、StoremanLottery.sol)的代码规范性、安全性以及业务逻辑三个方面进行多维度的安全审计。**经审计，该项目的智能合约未通过所有审计项，存在使用弃用项的问题，该问题为代码规范性问题，不会影响项目正常运行，合约可正常使用。**

代码规范审计

1. 编译器版本安全审计

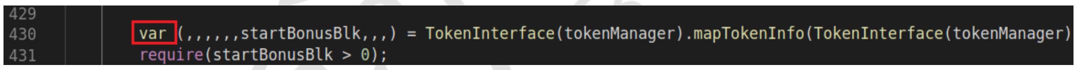
老版本的编译器可能会导致各种已知的安全问题，建议开发者在代码中指定合约代码采用最新的编译器版本，并消除编译器告警。

- **审计说明：**该项目智能合约指定了编译器最低版本为0.4.24，使用该版本的编译器编译时，无编译错误提示，但存在几处告警。
- **安全建议：该告警为代码规范告警，不存在安全隐患，但是仍建议修改代码，以消除告警。**
- **审计结果：通过**

2. 弃用项审计

Solidity智能合约开发语言处于快速迭代中，部分关键字已被新版本的编译器弃用，如throw、years等，为了消除其可能导致的隐患，合约开发者不应该使用当前编译器版本已弃用的关键字。

➢ **审计说明：**当前项目的合约指定最低编译器版本为0.4.24，该版本已弃用关键字var，但在项目合约中仍有多处使用了var关键字的情况(如图1)。



```
        var (,,,,,,startBonusBlk,,,) = TokenInterface(tokenManager).mapTokenInfo(TokenInterface(tokenManager)
        require(startBonusBlk > 0);
```

图1 使用var关键字部分源码

➢ **安全建议：建议将项目合约中使用var声明的变量替换为如下的方式：**

```
uint256 startBonusBlk;

( , , , , , , startBonusBlk , , , ) =

TokenInterface(tokenManager).mapTokenInfo(TokenInterface(tokenManager).mapKey(tokenOrigAddr));
```

➢ **修复结果：忽略。经项目方确认，无重大安全影响，暂不修复。**

➢ **审计结果：不通过**

3. 冗余代码审计

智能合约中的冗余代码会降低代码可读性，并可能需要消耗更多的gas用于合约部署，建议消除冗余代码。

➢ **审计说明：**在项目方提供的初始版本合约中出现了部分函数、结构体和变量声明但未使用的情况，详细情况如下表:

表1 项目合约声明但未被使用的函数、结构体和变量列表

所在合约	所在行数	冗余项名称	冗余情况说明
ImprovedStoremanGroupAdmin	79	TokenInfo	声明但未被使用
StoremanGroup	34	StakingPhase	声明但未被使用
StoremanGroup	44	locatedMpcAddr	未被赋值，一直为空
StoremanGroup	342	getCommisionPerUnit	内部函数，未被调用

➢ **安全建议：删除或修改代码中未使用的结构体、函数和变量；增加对locatedMpcAddr变量的赋值函数。**

➢ **修复结果：已在最新版合约代码中修复**

➢ **审计结果：通过**

4. require/assert 使用审计

Solidity使用状态恢复异常来处理错误。这种异常将会撤消对当前调用(及其所有子调用)中的状态所做的所有更改，并向调用者标记错误。函数assert和require可用于检查条件并在条件不满足时抛出异常。assert函数只能用于测试内部错误，并检查非变量。require函数用于确认条件有效性，例如输入变量，或合约状态变量是否满足条件，或验证外部合约调用的返回值。

- **审计结果：通过**
- **安全建议：无**

5. gas 消耗审计

以太坊虚拟机执行合约代码需要消耗gas，当gas不足时，代码执行会抛出out of gas异常，并撤销所有状态变更。合约开发者需要控制代码的gas消耗，避免因为gas不足导致函数执行一直失败。

- **审计说明：**项目合约中存在循环的操作，如确认抵押用户、发送乐透奖励等。但由于这些函数均是由owner调用，可合理控制输入数组参数的长度，来避免gas消耗过大。
- **审计结果：通过**
- **安全建议：无**

通用漏洞审计

1. 整型溢出审计

整型溢出是很多语言都存在的安全问题，它们在智能合约中尤其危险。Solidity最多能处理256位的数字(2**256-1)，最大数字增加1会溢出得到0。同样，当数字为uint类型时，0减去1会下溢得到最大数字值。溢出情况会导致不正确的结果，特别是如果其可能的结果未被预期，可能会影响程序的可靠性和安全性。

- **审计结果：通过**
- **安全建议：无**

2. 重入攻击审计

重入漏洞是最典型的以太坊智能合约漏洞，曾导致了The DAO被攻击。该漏洞原因是Solidity中的call.value()函数在被用来发送Ether的时候会消耗它接收到的所有gas，当调用call.value()函数发送Ether的逻辑顺序存在错误时，就会存在重入攻击的风险。

- **审计结果：通过**
- **安全建议：无**

3. 伪随机数生成审计

智能合约中可能会使用到随机数，在solidity下常见的是用block区块信息作为随机因子生成，但是这样使用是不安全的，区块信息是可以被矿工控制或被攻击者在交易时获取到，这类随机数在一定程度上是可预测或可碰撞的，比较典型的例子就是fomo3d的airdrop随机数可以被碰撞。

- **审计结果：通过**
- **安全建议：无**

4. 交易顺序依赖审计

在以太坊的交易打包执行过程中，面对相同难度的交易时，矿工往往会选择gas费用高的优先打包，因此用户可以指定更高的gas费用，使自己的交易优先被打包执行。

- **审计结果：通过**
- **安全建议：无**

5. 拒绝服务攻击审计

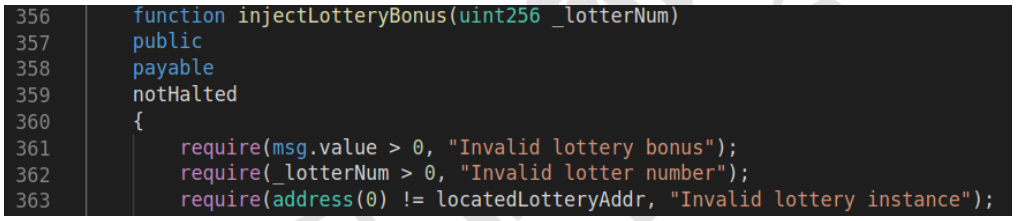
拒绝服务攻击，即Denial of Service，可以使目标无法提供正常的服务。在以太坊智能合约中也会存在此类问题，由于智能合约的不可更改性，该类攻击可能使得合约永远无法恢复正常工作状态。导致智能合约拒绝服务的原因有很多种，包括在作为交易接收方时的恶意revert、代码设计缺陷导致gas耗尽等等。

- **审计结果：通过**
- **安全建议：无**

6. 函数调用权限审计

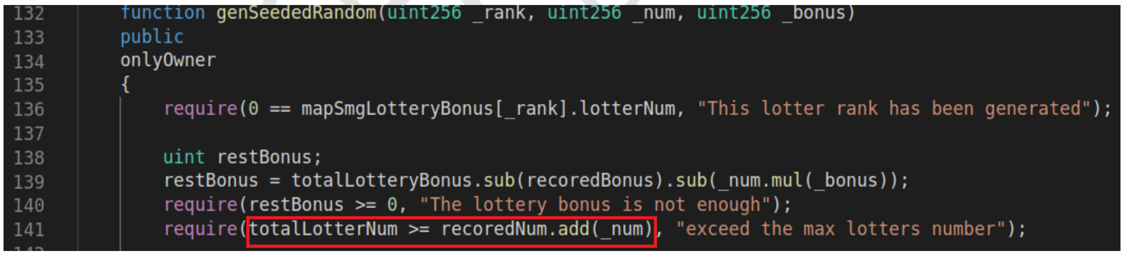
智能合约如果存在高权限功能，如：铸币、自毁、change owner等，需要对函数调用做权限限制，避免权限泄露导致的安全问题。

- **审计说明**：在项目方提供的初始版本合约代币中，StoremanGroup合约的injectLotteryBonus函数存在着缺少调用限制的安全问题。如图2所示，任何人可通过调用injectLotteryBonus函数修改StoremanLottery合约的“totalLotterNum”参数。特别的，如果将“totalLotterNum”参数修改为较小的值时，可能出现“totalLotterNum < recoredNum.add(_num)”而导致不能通过如图3红框所示的检查，使得合约不能正常生成随机数。



```
    function injectLotteryBonus(uint256 _lotterNum)
    public
    payable
    notHalted
    {
        require(msg.value > 0, "Invalid lottery bonus");
        require(_lotterNum > 0, "Invalid lotter number");
        require(address(0) != locatedLotteryAddr, "Invalid lottery instance");
```

图2 injectLotteryBonus函数部分源代码截图



```
    function genSeededRandom(uint256 _rank, uint256 _num, uint256 _bonus)
    public
    onlyOwner
    {
        require(0 == mapSmgLotteryBonus[_rank].lotterNum, "This lotter rank has been generated");

        uint restBonus;
        restBonus = totalLotteryBonus.sub(recoredBonus).sub(_num.mul(_bonus));
        require(restBonus >= 0, "The lottery bonus is not enough");
        require(totalLotterNum >= recoredNum.add(_num), "exceed the max lotters number");
```

图3 genSeededRandom函数部分源代码截图

- **安全建议：修改上述函数的调用权限为owner。**

- **审计结果：通过。上述问题已在最新版本合约代码中修复。**

7. call/delegatecall 安全审计

Solidity 中提供了call/delegatecall函数来进行函数调用，如果使用不当，会造成call注入漏洞，例如call的参数如果可控，则可以控制本合约进行越权操作或调用其他合约的危险函数。

- **审计结果：通过**

- **安全建议：无**

8. 返回值安全审计

在Solidity中存在transfer()、send()、call.value()等方法中，transfer转账失败交易会回滚，而send和call.value转账失败会return false，如果未对返回做正确判断，则可能会执行到未预期的逻辑;另外在ERC20 Token的transfer/transferFrom功能实现中，也要避免转账失败return false的情况，以免造成假充值漏洞。

- **审计结果：通过**

- **安全建议：无**

9. tx.origin使用安全审计

在以太坊智能合约的复杂调用中，tx.origin表示交易的初始发起者地址，如果使用tx.origin进行权限判断，可能会出现错误;另外，如果合约需要判断调用方是否为合约地址时则需要使用tx.origin，不能使用extcodesize。

- **审计结果：通过**
- **安全建议：无**

10. 重放攻击审计

重放攻击是指如果两份合约使用了相同的代码实现，并且身份鉴权在传参中，当用户在向一份合约中执行一笔交易，交易信息可以被复制并且向另一份合约重放执行该笔交易。

- **审计结果：通过**
- **安全建议：无**

11. 变量覆盖审计

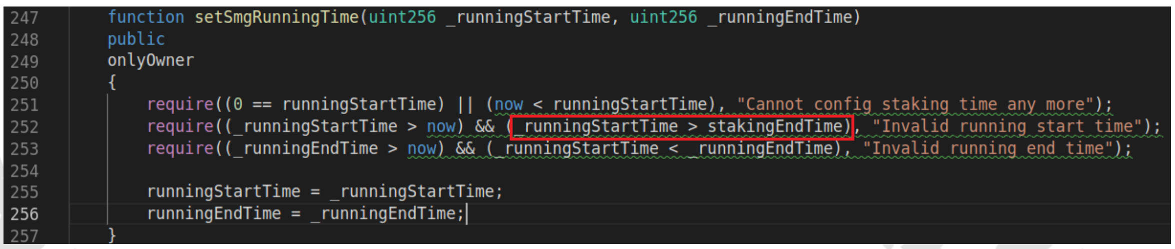
以太坊存在着复杂的变量类型，例如结构体、动态数组等，如果使用不当，对其赋值后，可能导致覆盖已有状态变量的值，造成合约执行逻辑异常。

- **审计结果：通过**
- **安全建议：无**

业务审计

1. 可能会因管理员误操作，导致合约状态异常

- **问题描述：**对于StoremanGroup合约正常业务逻辑的执行顺序应该是：先开放抵押功能，然后owner利用获取的抵押金去注册smg，因此抵押阶段的开放时间应该在活动开放时间之前。但在初始版本合约中，如果先调用setSmgRunningTime函数，后调用setSmgStakingTime函数，在设置RunningTime时，stakingEndTime仍为0(如图4所示，stakingEndTime默认为0)，任意指定活动开始时间都成立。因此，最终结果可能出现"runningStartTime<stakingEndTime"的情况，导致后面的业务逻辑无法正常执行。



```
    function setSmgRunningTime(uint256 _runningStartTime, uint256 _runningEndTime)
    public
    onlyOwner
    {
        require((0 == runningStartTime) || (now < runningStartTime), "Cannot config staking time any more");
        require((_runningStartTime > now) && (_runningStartTime > stakingEndTime), "Invalid running start time");
        require((_runningEndTime > now) && (_runningStartTime < _runningEndTime), "Invalid running end time");

        runningStartTime = _runningStartTime;
        runningEndTime = _runningEndTime;
    }
```

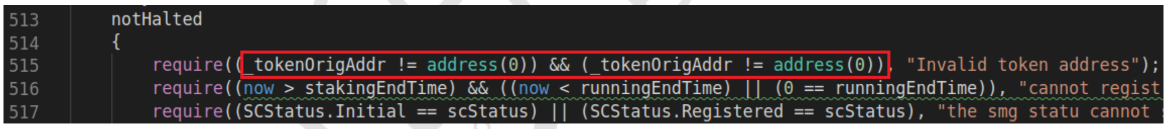
图4 setSmgRunningTime函数源代码截图

- **安全建议：该问题可线下指定调用顺序，或者在setSmgRunningTime函数中增加"stakingEndTime != 0"的检查。**

➢ **修复结果：已修复。已在最新版合约代码中添加“0 != stakingEndTime”的检查。**

2. 代码书写错误

➢ **问题描述：**如图5所示，根据业务逻辑，该行的代码应该是“require((_tokenOrigAddr != address(0)) && (_originalChainAddr != address(0)), "Invalid token address");”。



```
    notHalted
    {
        require((_tokenOrigAddr != address(0)) && (_tokenOrigAddr != address(0)), "Invalid token address");
        require((now > stakingEndTime) && ((now < runningEndTime) || (0 == runningEndTime)), "cannot regist
        require((SCStatus.Initial == scStatus) || (SCStatus.Registered == scStatus), "the smg statu cannot
```

图5 applyRegisterSmgToAdmin函数部分源代码截图

➢ **安全建议：修改代码**

➢ **修复结果：已修复**

合约源代码审计注释：

// 成都链安 // ###### ImprovedStoremanGroupAdmin 合约 ######

```
/*

  Copyright 2018 Wanchain Foundation.

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.

*/

//
//                                  _           _           _
//  __      ____ _ _ __   ___| |__   __ _(_)_ __   __| | _____   __
//  \ \ /\ / / _` | '_ \ / __| '_ \ / _` | | '_ \@/ _` |/ _ \ \ / /
//   \ V  V / (_| | | | | (__| | | | (_| | | | | | (_| |  __/\ V /
//    \_/\_/ \__,_|_| |_|\___|_| |_|\__,_|_|_| |_|\__,_|\___| \_/
//
//  Code style according to: https://github.com/wanchain/wanchain-
token/blob/master/style-guide.rst
```

```
pragma solidity ^0.4.24; // 成都链安 // 建议固定编译器版本，并消除编译器告警

import "./SafeMath.sol";
import "./Halt.sol";
// 成都链安 // TokenManager 合约接口，原 StoremanGroupAdmin 合约可通过该合约适配
TokenManager 合约
// 成都链安 // 建议接口函数使用“external”声明函数可见性
interface TokenInterface {
    function mapKey(address) public view returns(bytes32);
    function mapTokenInfo(bytes32) public view returns(address, address, uint, uint,
uint, bool, uint, uint, uint, uint);
    function mapPenaltyReceiver(address) public view returns(address);
    function updateTotalBonus(address, uint, bool) external returns(bool);
    function DEFAULT_PRECISE() public returns (uint); // 成都链安 // 建议添加“view”关
键字
}
// 成都链安 // QuotaLedger 合约接口，原 StoremanGroupAdmin 合约可通过该合约适配
QuotaLedger 合约
interface QuotaInterface {
    function applyUnregistration(address, address) external returns (bool);
    function setStoremanGroupQuota(address, address, uint) external returns (bool);
    function unregisterStoremanGroup(address, address, bool) external returns
(bool);
}
// 成都链安 // WanToken 合约接口，包含 decimals 接口函数，建议接口函数使用“external”声明
函数可见性，并添加“view”关键字
interface WERCProtocol {
    function decimals() public returns(uint8);
}

// modified by liaoxx on 20181228, for Improved StoremanGroupAdmin task
contract ImprovedStoremanGroupAdmin is Halt{
    using SafeMath for uint; // 成都链安 // 引用 SafeMath 库，用于安全数学运算

    /// token manager instance address
    address public tokenManager;

    /// quotaLedger instance address
    address public quotaLedger;

    /// added by liaoxx on 2018.12.28, for oldStoremanGroupAdminArray instance
address
    mapping(address=>bool) public mapRegistedOldSmgAdmin;
    /// a map from addresses to storeman group information (tokenOrigAddr->storemanGroupAddr->StoremanGroup)
    mapping(address=>mapping(address => StoremanGroup)) public mapStoremanGroup;

    /// a map from addresses to storeman group white list information
    mapping(address=>mapping(address => bool)) public mapSmgWhiteList;
    // 成都链安 // smg 结构体，存储 smg 信息
    struct StoremanGroup {
        uint    deposit;                  /// the storeman group deposit in wan coins
        address originalChainAddr;        /// the account for storeman group on original chain
        uint    unregisterApplyTime;      /// the time point for storeman group applied unregistration
        uint    txFeeRatio;               /// the fee ratio required by storeman group
        uint    bonusBlockNumber;         /// the start block number for bonus calculation for storeman group
        address initiator;                /// the account for registering a storeman group which provides storeman group deposit
        uint    punishPercent;            /// punish rate of deposit, which is an integer from 0 to 100
    }

    /**
     *
     * EVENTS
     *
     */

    /// @notice                           event for storeman register
    /// @dev                              event for storeman register
    /// @param tokenOrigAddr              token address of original chain
    /// @param smgWanAddr                 smgWanAddr address
    /// @param smgOrigAddr                storeman group  original chain address
    /// @param wanDeposit                 deposit wancoin number
    /// @param quota                      corresponding token quota
    /// @param txFeeRatio                 storeman fee ratio
    event StoremanGroupRegistrationLogger(address indexed tokenOrigAddr, address indexed smgWanAddr, address smgOrigAddr, uint wanDeposit, uint quota, uint txFeeRatio);

    /// @notice                           event for bonus deposit
    /// @dev                              event for bonus deposit
    /// @param tokenOrigAddr              token address of original chain
    /// @param sender                     sender for bonus
    /// @param wancoin                    deposit wancoin number
    event StoremanGroupDepositBonusLogger(address indexed tokenOrigAddr, address
indexed sender, uint indexed wancoin);

    /// @notice                         event for storeman register
    /// @dev                            event for storeman register
    /// @param smgWanAddr               storeman address
    /// @param tokenOrigAddr            token address of original chain
    event SmgEnableWhiteListLogger(address indexed smgWanAddr, address indexed
tokenOrigAddr);

    /// @notice                         event for applying storeman group
unregister
    /// @param tokenOrigAddr            token address of original chain
    /// @param smgWanAddr               storemanGroup address
    /// @param applyTime                the time for storeman applying unregister
    event StoremanGroupApplyUnRegistrationLogger(address indexed tokenOrigAddr,
address indexed smgWanAddr, uint indexed applyTime);

    /// @notice                         event for storeman group withdraw deposit
    /// @param tokenOrigAddr            token address of original chain
    /// @param smgWanAddr               storemanGroup address
    /// @param actualReturn             the time for storeman applying unregister
    /// @param deposit                  deposit in the first place
    event StoremanGroupWithdrawLogger(address indexed tokenOrigAddr, address indexed
smgWanAddr, uint indexed actualReturn, uint deposit);

    /// @notice                         event for storeman group claiming system
bonus
    /// @param tokenOrigAddr            token address of original chain
    /// @param bonusRecipient           storemanGroup address
    /// @param bonus                    the bonus for storeman claim
    event StoremanGroupClaimSystemBonusLogger(address indexed tokenOrigAddr, address
indexed bonusRecipient, uint indexed bonus);


    /// @dev  only registed Old SmgAdmin can call this function
    /// modifier
    modifier onlyRegistedOldSmgAdmin() {
        require(true == mapRegistedOldSmgAdmin[msg.sender]); // 成都链安 // 要求被修饰函
数的调用者必须已注册的原 StoremanGroupAdmin 合约
        _;
    }

    /**
     *
     * private methods
     *
     */
    /// @notice                              set oldStoremanGroupAdmin instance address
added by liaoxx for
    /// @param _smga                         old StoremanGroupAdmin instance address
    // 成都链安 // 注册原 StoremanGroupAdmin 合约地址(仅在 halted 为 true 时，可由 owner 调
用)
    function registerOldStoremanGroupAdmin(address _smga)
        public
        onlyOwner
        isHalted
    {
        require(_smga != address(0), "Invalid storemangroup Address"); // 成都链安 //
参数_smga 非零地址检查
        mapRegistedOldSmgAdmin[_smga] = true; // 成都链安 // 注册_smga 为原
StoremanGroupAdmin 合约地址
    }


    /// @notice                              function for bonus claim
    /// @param tokenOrigAddr                 token address of original chain
    /// @param storemanGroup                 storeman group address
    // 成都链安 // 领取系统周期奖励
    function doClaimSystemBonus(address tokenOrigAddr, address storemanGroup)
        private
    {
        StoremanGroup storage smgInfo =
mapStoremanGroup[tokenOrigAddr][storemanGroup];
        // 成都链安 // 获取对应代币的部分信息
        // 成都链安 // var 关键字已弃用，建议指明变量类型
        var (,,,,,,startBonusBlk,bonusTotal,bonusPeriodBlks,bonusRatio) =
TokenInterface(tokenManager).mapTokenInfo(TokenInterface(tokenManager).mapKey(tokenO
rigAddr));

        //if(smgInfo.punishPercent != 0 || startBonusBlk == 0){  // modified by
liaoxx On 2019.01.07
        if(startBonusBlk == 0){
          return; // 成都链安 // 如果该 smg 对应的代币未参与周期奖励，则直接返回
        }
        // 成都链安 // 设置 smg 参与周期奖励的起始区块号
        smgInfo.bonusBlockNumber = smgInfo.bonusBlockNumber < startBonusBlk ?
startBonusBlk : smgInfo.bonusBlockNumber;
        // 成都链安 // 当该 smg 参与周期奖励持续时间不低于一个周期，且拥有资金大于 0 时，才能
获取周期奖励
        if (block.number.sub(smgInfo.bonusBlockNumber) >= bonusPeriodBlks &&
smgInfo.deposit > 0) {
            uint cycles =
(block.number.sub(smgInfo.bonusBlockNumber)).div(bonusPeriodBlks); // 成都链安 // 获取参
与奖励的周期数
            // 成都链安 // 更新该 smg 参与周期奖励的起始区块号
            smgInfo.bonusBlockNumber =
smgInfo.bonusBlockNumber.add(cycles.mul(bonusPeriodBlks));
            // 成都链安 // 计算一个周期应得奖励金额
            uint bonus =
smgInfo.deposit.mul(bonusRatio).div(TokenInterface(tokenManager).DEFAULT_PRECISE());
            bonus = bonus.mul(cycles); // 成都链安 // 计算该 smg 应得总奖励金额
            // 成都链安 // 当 smg 应得总奖励金额大于 0，但不大于剩余总奖金时，可领取应得奖励
            if (bonusTotal >= bonus && bonus > 0) {
                require(TokenInterface(tokenManager).updateTotalBonus(tokenOrigAddr,
bonus, false)); // 成都链安 // 更新系统周期奖励额度
                    if(smgInfo.initiator != address(0)){ // 成都链安 // 如果该 smg 存在
代理人，将奖励发送至代理人
                        smgInfo.initiator.transfer(bonus);
                        emit StoremanGroupClaimSystemBonusLogger(tokenOrigAddr,
smgInfo.initiator, bonus);
                    } else { // 成都链安 // 如果该 smg 不存在代理人，则将奖励发送给该 smg
                        msg.sender.transfer(bonus);
                        emit StoremanGroupClaimSystemBonusLogger(tokenOrigAddr,
msg.sender, bonus);
                    }
            } else { // 成都链安 // 如果该 smg 不存在可领取的奖励，将直接触发领取奖励为 0 的
事件
                emit StoremanGroupClaimSystemBonusLogger(tokenOrigAddr, msg.sender,
0);
            }
        }
    }

    /**
    *
    * MANIPULATIONS
    *
    */

    /// @notice                     set tokenManager instance address and
quotaLedger instance address
    /// @param tm                   token manager instance address
    /// @param ql                   quota ledger instance address
    // 成都链安 // 设置 QuotaLedger、TokenManager 合约地址(仅在 halted 为 true 时，可由
owner 调用)
    function injectDependencies(address tm, address ql)
```

```
        public
        onlyOwner
        isHalted
    {
        require(tm != address(0) && ql != address(0)); // 成都链安 // 地址参数非零检查
        // 成都链安 // 设置 QuotaLedger、TokenManager 合约地址
        tokenManager = tm;
        quotaLedger = ql;
    }


    /// @notice                       function for setting smg white list by
owner
    /// @param tokenOrigAddr          token address of original chain
    /// @param storemanGroup          storemanGroup address for whitelist
    // 成都链安 // 设置 smg 白名单(仅可由 owner 调用)
    function setSmgWhiteList(address tokenOrigAddr, address storemanGroup)
        public
        onlyOwner
    {
        require(storemanGroup!=address(0)); // 成都链安 // 输入参数 storemanGroup 非零地
址检查

        // make sure white list mechanism is enabled
        // 成都链安 // 获取对应代币是否开启白名单
        // 成都链安 // var 关键字已弃用，建议指明变量类型
        var (,,,,,useWhiteList,,,,) =
TokenInterface(tokenManager).mapTokenInfo(TokenInterface(tokenManager).mapKey(tokenO
rigAddr));
        require(useWhiteList); // 成都链安 // 要求该代币已开启白名单

        require(mapStoremanGroup[tokenOrigAddr][storemanGroup].bonusBlockNumber ==
0); // 成都链安 // 要求该 smg 未注册
        require(!mapSmgWhiteList[tokenOrigAddr][storemanGroup]); // 成都链安 // 要求该
smg 未加入白名单

        mapSmgWhiteList[tokenOrigAddr][storemanGroup] = true; // 成都链安 // 将该 smg 加
入白名单

        emit SmgEnableWhiteListLogger(storemanGroup, tokenOrigAddr);
    }

    /// @notice                       function for storeman registration, this
method should be invoked by the storemanGroup himself
    /// @param tokenOrigAddr          token address of original chain
    /// @param originalChainAddr          the storeman group info on original chain
    /// @param txFeeRatio                 the transaction fee required by storeman
group
    // 成都链安 // smg 注册(仅在 halted 为 false 时可调用)
    function storemanGroupRegister(address tokenOrigAddr, address originalChainAddr,
uint256 txFeeRatio)
        public
        payable
        notHalted
    {
        storemanGroupRegisterByDelegate(tokenOrigAddr, msg.sender,
originalChainAddr, txFeeRatio); // 成都链安 // 调用 storemanGroupRegisterByDelegate 函
数注册 smg
    }

    /// @notice                           function for storeman register by sender
this method should be
    ///                                   invoked by a storemanGroup registration
proxy or wanchain foundation
    /// @param tokenOrigAddr              token address of original chain
    /// @param storemanGroup              the storeman group register address
    /// @param originalChainAddr          the storeman group info on original chain
    /// @param txFeeRatio                 the transaction fee required by storeman
group
    // 成都链安 // smg 代理注册(仅在 halted 为 false 时调用)
    function storemanGroupRegisterByDelegate(address tokenOrigAddr, address
storemanGroup, address originalChainAddr,uint txFeeRatio)
        public
        payable
        notHalted
    {
        // 成都链安 // smg 注册需满足：(1)对应的 smg 地址和原链地址不为空；(2)设置的手续费率
大于 0；(3)该 smg 未注册
        require(storemanGroup != address(0) && originalChainAddr != address(0) &&
txFeeRatio > 0);
        require(mapStoremanGroup[tokenOrigAddr][storemanGroup].deposit == uint(0));
        // 成都链安 // 获取对应代币的部分信息
        // 成都链安 // var 关键字已弃用，建议指明变量类型
        var (,tokenWanAddr,token2WanRatio,minDeposit,,useWhiteList,,,,) =
TokenInterface(tokenManager).mapTokenInfo(TokenInterface(tokenManager).mapKey(tokenO
rigAddr));
        // 成都链安 // smg 注册需满足：(4)注册所携带的资金不小于该代币的最小投资额度
        require(msg.value >= minDeposit &&
mapStoremanGroup[tokenOrigAddr][storemanGroup].bonusBlockNumber == 0);

        // white list filter
        if (useWhiteList) {
            require(mapSmgWhiteList[tokenOrigAddr][storemanGroup]); // 成都链安 // 如果其对应代币开启了白名单，则需检查该 smg 地址是否已加入白名单
            mapSmgWhiteList[tokenOrigAddr][storemanGroup] = false; // 成都链安 // 更新代币白名单中该地址状态，避免二次注册
        }
        // 成都链安 // 计算注册时该 smg 可获得的代币额度
        uint quota = 
(msg.value).mul(TokenInterface(tokenManager).DEFAULT_PRECISE()).div(token2WanRatio).mul(10**uint(WERCProtocol(tokenWanAddr).decimals())).div(1 ether);

        // regsiter this storeman group with calculated quota
        // 成都链安 // 调用 QuotaLedger 合约 setStoremanGroupQuota 函数设置该 smg 的额度
        require(QuotaInterface(quotaLedger).setStoremanGroupQuota(tokenOrigAddr, storemanGroup, quota));
        // 成都链安 // 创建 smg
        mapStoremanGroup[tokenOrigAddr][storemanGroup] = StoremanGroup(msg.value, originalChainAddr, 0, txFeeRatio, block.number, storemanGroup == msg.sender ? address(0) : msg.sender, 0);

        /// fire event
        emit StoremanGroupRegistrationLogger(tokenOrigAddr, storemanGroup, originalChainAddr, msg.value, quota,txFeeRatio);
    }

    /// @notice                         function for storemanGroup applying unregister
    /// @dev                            function for storemanGroup applying unregister
    /// @param tokenOrigAddr            token address of original chain
    // 成都链安 // smg 申请注销(仅在 halted 为 false 时，可被调用)
    function storemanGroupApplyUnregister(address tokenOrigAddr)
        public
        notHalted
    {
        smgApplyUnregisterByDelegate(tokenOrigAddr, msg.sender); // 成都链安 // 调用 smgApplyUnregisterByDelegate 函数申请注销
    }

    /// @notice                         apply unregistration through a proxy
    /// @dev                            apply unregistration through a proxy
    /// @param tokenOrigAddr            token address of original chain
    /// @param storemanGroup            storemanGroup address
    // 成都链安 // smg 代理申请注销(仅在 halted 为 false 时，可被调用)
    function smgApplyUnregisterByDelegate(address tokenOrigAddr, address
storemanGroup)
        public
        notHalted
    {
        if (msg.sender != storemanGroup) { // 成都链安 // 要求该函数调用者是输入的 smg 或
其代理人
            require(mapStoremanGroup[tokenOrigAddr][storemanGroup].initiator ==
msg.sender);
        }

        // make sure this storemanGroup has registered
        StoremanGroup storage smg = mapStoremanGroup[tokenOrigAddr][storemanGroup];
        // 成都链安 // smg 申请注销需满足：(1)该 smg 已注册；(2)该 smg 未申请注销
        require(smg.bonusBlockNumber > 0);
        // make sure this storemanGroup has not applied
        require(smg.unregisterApplyTime == 0);

        smg.unregisterApplyTime = now; // 成都链安 // 更新申请注销时间戳
        // 成都链安 // var 关键字已弃用，建议指明变量类型
        var (,,,,,,startBonusBlk,,,) =
TokenInterface(tokenManager).mapTokenInfo(TokenInterface(tokenManager).mapKey(tokenO
rigAddr));

        //if (startBonusBlk > 0 && smg.punishPercent==0) // modified by liaoxx On
2019.01.07
        if (startBonusBlk > 0) {
            doClaimSystemBonus(tokenOrigAddr, storemanGroup); // 成都链安 // 如果对应的
代币参与了周期奖励，则调用 doClaimSystemBonus 函数领取周期奖励
        }
        // 成都链安 // 调用 QuotaLedger 合约 applyUnregistration 函数申请注销，并检查调用结
果
        require(QuotaInterface(quotaLedger).applyUnregistration(tokenOrigAddr,
storemanGroup));

        // fire event
        emit StoremanGroupApplyUnRegistrationLogger(tokenOrigAddr, storemanGroup,
now);
    }

    /// @notice                     function for storeman group withdraw
deposit
    /// @dev                        function for storeman group withdraw
deposit
    /// @param tokenOrigAddr        token address of original chain
    // 成都链安 // smg 退款(仅在 halted 为 false 时，可被调用)
    function storemanGroupWithdrawDeposit(address tokenOrigAddr)
        public
        notHalted
    {
        smgWithdrawDepositByDelegate(tokenOrigAddr, msg.sender); // 成都链安 // 调用
smgWithdrawDepositByDelegate 函数进行退款
    }

    /// @notice                     withdraw deposit through a proxy
    /// @dev                        withdraw deposit through a proxy
    /// @param tokenOrigAddr        token address of original chain
    /// @param storemanGroup        storemanGroup address
    // 成都链安 // smg 代理退款(仅在 halted 为 false 时，可被调用)
    function smgWithdrawDepositByDelegate(address tokenOrigAddr, address
storemanGroup)
        public
        notHalted
    {
        if (msg.sender != storemanGroup) { // 成都链安 // 要求该函数调用者是输入的 smg 或
其代理人
            require(mapStoremanGroup[tokenOrigAddr][storemanGroup].initiator ==
msg.sender);
        }

        StoremanGroup storage smg = mapStoremanGroup[tokenOrigAddr][storemanGroup];
        // 成都链安 // var 关键字已弃用，建议指明变量类型
        var (,,,,withdrawDelayTime,,,,,) =
TokenInterface(tokenManager).mapTokenInfo(TokenInterface(tokenManager).mapKey(tokenO
rigAddr));
        // 成都链安 // 退款需满足：(1)距申请注销的时间差已超过退款延迟时间；(2)该 smg 已注册
        require(now > smg.unregisterApplyTime.add(withdrawDelayTime) &&
smg.deposit > 0);
        // 成都链安 // 调用 QuotaLedger 合约中的 unregisterStoremanGroup 函数注销 smg，并
检查调用结果
        require(QuotaInterface(quotaLedger).unregisterStoremanGroup(tokenOrigAddr,
storemanGroup, true));
        // 成都链安 // 获取该 smg 的注册资金
        uint deposit = smg.deposit;
        uint restBalance = smg.deposit;

        // modified by liaoxx On 2019.01.07
        /*
        if (smg.punishPercent > 0) {
            // transfer penalty to the penaltyReceiver of corresponding ERC20 token
            restBalance =
restBalance.sub(restBalance.mul(smg.punishPercent).div(100));
            address penaltyReceiver =
TokenInterface(tokenManager).mapPenaltyReceiver(tokenOrigAddr);
            require(penaltyReceiver != address(0));
            penaltyReceiver.transfer(deposit.sub(restBalance));
        }
        */
        // 成都链安 // 清空 smg 数据
        smg.deposit = 0;
        smg.originalChainAddr = address(0);
        smg.unregisterApplyTime = 0;
        smg.txFeeRatio = 0;
        smg.bonusBlockNumber = 0;
        smg.punishPercent = 0;

        if (smg.initiator != address(0)) { // 成都链安 // 如果该 smg 存在代理人，退款给代
理人
            smg.initiator.transfer(restBalance);
        } else { // 成都链安 // 如果该 smg 不存在代理人，则退款给 smg
            msg.sender.transfer(restBalance);
        }
        smg.initiator = address(0); // 成都链安 // 清空 smg 代理人信息
        // 成都链安 // 建议在 smg 清空所有数据后，执行 delete，移除该 smg
        emit StoremanGroupWithdrawLogger(tokenOrigAddr, storemanGroup, restBalance,
deposit);
    }

    /// @notice                     function for storeman claiming system
bonus
    /// @dev                        function for storeman claiming system
bonus
    /// @param tokenOrigAddr        token address of original chain
    // 成都链安 // 领取系统周期奖励(仅在 halted 为 false 时，可被 smg 调用)
    function storemanGroupClaimSystemBonus(address tokenOrigAddr)
        public
        notHalted
    {
        StoremanGroup storage smg = mapStoremanGroup[tokenOrigAddr][msg.sender];
        // 成都链安 // 领取系统周期奖励需满足：(1)该 smg 已注册；(2)该 smg 未申请注销
        require(smg.bonusBlockNumber != 0 && smg.unregisterApplyTime == 0);
        // 成都链安 // 调用 doClaimSystemBonus 函数领取系统周期奖励
        doClaimSystemBonus(tokenOrigAddr, msg.sender);
    }

    /// @notice                     function for storeman claiming system
bonus through a proxy
```

```
    /// @dev                          function for storeman claiming system
bonus through a proxy
    /// @param tokenOrigAddr          token address of original chain
    /// @param storemanGroup          storemanGroup address
    // 成都链安 // 代理领取系统周期奖励(仅在 halted 为 false 时，可被调用)
    function smgClaimSystemBonusByDelegate(address tokenOrigAddr, address
storemanGroup)
        public
        notHalted
    {
        // make sure the address who registered this smg initiated this transaction
        // 成都链安 // 代理领取系统周期奖励需满足：(1)函数调用者为该 smg 的代理人；(2)该 smg
已注册；(3)该 smg 未申请注销
        require(mapStoremanGroup[tokenOrigAddr][storemanGroup].initiator ==
msg.sender);

        StoremanGroup storage smg = mapStoremanGroup[tokenOrigAddr][storemanGroup];
        require(smg.bonusBlockNumber != 0 && smg.unregisterApplyTime == 0);
        // 成都链安 // 调用 doClaimSystemBonus 函数领取系统周期奖励
        doClaimSystemBonus(tokenOrigAddr, storemanGroup);
    }

    /// @notice                       function for bonus deposit
    /// @dev                          function for bonus deposit
    /// @param tokenOrigAddr          token address of original chain
    // 成都链安 // 存入 smg 奖励资金(仅可由 owner 调用)
    function depositSmgBonus(address tokenOrigAddr)
        public
        payable
        onlyOwner
    {
        require(msg.value > 0); // 成都链安 // 要求存入的资金大于 0
        // 成都链安 // var 关键字已弃用，建议指明变量类型
        var (,,,,,,startBonusBlk,,,) =
TokenInterface(tokenManager).mapTokenInfo(TokenInterface(tokenManager).mapKey(tokenO
rigAddr));
        require(startBonusBlk > 0); // 成都链安 // 要求其对应的代币已经开启周期奖励

        require(TokenInterface(tokenManager).updateTotalBonus(tokenOrigAddr,
msg.value, true)); // 成都链安 // 增加周期奖励总额度

        emit StoremanGroupDepositBonusLogger(tokenOrigAddr, msg.sender, msg.value);
    }

    /// @notice function for destroy contract
    // 成都链安 // 自毁函数(仅在 halted 为 true 时，可被 owner 调用)
    function kill()
        public
        isHalted
        onlyOwner
    {
        selfdestruct(owner); // 成都链安 // 销毁合约，并将合约余额发送至 owner
    }

    /// @notice fallback function
    function () public payable {
       revert();
    }

    // functions below as agent interface for old version storemanGroupAdmin
instance, added by liaoxx On 2018.12.28
    /// @notice                     agent function for oldStoremanGroupAdmin
to get keccak256 hash key of Token
    /// @param tokenOrigAddr        token address of original chain
    // 成都链安 // 获取指定代币的 key(仅可由原 StoremanGroupAdmin 合约调用)
    function mapKey(address tokenOrigAddr)
    public
    view
    onlyRegistedOldSmgAdmin
    returns(bytes32)
    {
        //require(mapRegistedOldSmgAdmin[msg.sender], "It is not an valid old
storeman group admin object");
        require(tokenOrigAddr != address(0), "Invalid token Origin Address"); // 成都
链安 // 对应的代币地址非零检查
        bytes32 keyHash = TokenInterface(tokenManager).mapKey(tokenOrigAddr); // 成都
链安 // 调用 TokenManager 合约的 mapKey 函数获取其 key
        return keyHash;
    }

    /// @notice                     agent function for oldStoremanGroupAdmin
to get TokenInfo from TokenManager
    /// @param keyHash              token keccak256 hash key
    // 成都链安 // 获取 keyHash 所对应的代币信息(仅可由原 StoremanGroupAdmin 合约调用)
    function mapTokenInfo(bytes32 keyHash)
    public
    view
    onlyRegistedOldSmgAdmin
    returns(address, address, uint, uint, uint, bool, uint, uint, uint, uint)
    {
        //require(mapRegistedOldSmgAdmin[msg.sender], "It is not an valid old
storeman group admin object");
        require(keyHash != bytes32(0), "Invalid key hash"); // 成都链安 // 要求输入的
keyHash 不为空

        return TokenInterface(tokenManager).mapTokenInfo(keyHash); // 成都链安 // 调用
TokenManager 合约的 mapTokenInfo 函数，返回对应代币信息
    }

    /// @notice                            agent function for oldStoremanGroupAdmin
to get penalty receiver address from TokenManager
    /// @param tokenOrigAddr               token address of original chain
    // 成都链安 // 获取指定代币的罚金接受者(仅可由原 StoremanGroupAdmin 合约调用)
    function mapPenaltyReceiver(address tokenOrigAddr)
    public
    view
    onlyRegistedOldSmgAdmin
    returns(address)
    {
        //require(mapRegistedOldSmgAdmin[msg.sender], "It is not an valid old
storeman group admin object");
        require(tokenOrigAddr != address(0), "Invalid token Origin Address"); // 成都
链安 // 对应代币地址非零检查

        address receiver =
TokenInterface(tokenManager).mapPenaltyReceiver(tokenOrigAddr); // 成都链安 // 调用
TokenManager 合约的 mapPenaltyReceiver 函数，获取其代币的罚金接受者
        ///require(penaltyReceiver != address(0));

        return receiver;
    }

    /// @notice                              agent function for oldStoremanGroupAdmin
to get penalty receiver address from TokenManager
    /// @param tokenOrigAddr                 token address of original chain
    /// @param bonus                         bonus value
    /// @param isAdded                       plus if true, else do a minus operation
    // 成都链安 // 更新指定代币的系统周期奖励总额度(仅可由原 StoremanGroupAdmin 合约调用)
    function updateTotalBonus(address tokenOrigAddr, uint bonus, bool isAdded)
    external
    onlyRegistedOldSmgAdmin
    returns(bool)
    {
        //require(mapRegistedOldSmgAdmin[msg.sender], "It is not an valid old
storeman group admin object");
        require(tokenOrigAddr != address(0), "Invalid token Origin Address"); // 成都
```

```
链安 // 对应代币地址非零检查
        // 成都链安 // 调用 TokenManager 合约的 updateTotalBonus 函数，更新系统周期奖励总
额度
        bool ifUpdated =
TokenInterface(tokenManager).updateTotalBonus(tokenOrigAddr, bonus, isAdded);

        return ifUpdated;
    }

    /// @notice                              agent function for oldStoremanGroupAdmin
to get DEFAULT_PRECISE from TokenManager
    // 成都链安 // 获取默认精度(仅可由原 StoremanGroupAdmin 合约调用)
    // 成都链安 // 建议添加“view”关键字
    function DEFAULT_PRECISE()
    public
    onlyRegistedOldSmgAdmin
    returns(uint)
    {
        //require(mapRegistedOldSmgAdmin[msg.sender], "It is not an valid old
storeman group admin object");
        // 成都链安 // 调用 TokenManager 合约的 DEFAULT_PRECISE 函数，获取默认精度
        uint defaultPrecise = TokenInterface(tokenManager).DEFAULT_PRECISE();

        return defaultPrecise;
    }

    /// @notice                              agent function for oldStoremanGroupAdmin
to apply unregister to quotaLedger
    /// @param tokenOrigAddr                 token address of original chain
    /// @param storemanGroup                 storemanGroup address
    // 成都链安 // 申请注销 smg(仅可由原 StoremanGroupAdmin 合约调用)
    function applyUnregistration(address tokenOrigAddr, address storemanGroup)
    external
    onlyRegistedOldSmgAdmin
    returns (bool)
    {
        //require(mapRegistedOldSmgAdmin[msg.sender], "It is not an valid old
storeman group admin object");
        // 成都链安 // 调用 QuotaLedger 合约的 applyUnregistration 函数，申请注销 smg
        bool ifSucc = QuotaInterface(quotaLedger).applyUnregistration(tokenOrigAddr,
storemanGroup);

        return ifSucc;
    }

    /// @notice                              gent function for oldStoremanGroupAdmin
```

```
to set storeman group's quota
    /// @param tokenOrigAddr                    token address of original chain
    /// @param storemanGroup                    storemanGroup address
    /// @param quota                            a storemanGroup's quota
```
// 成都链安 // 设置 smg 的额度(仅可由原 StoremanGroupAdmin 合约调用。目前直接返回 false)
```
    function setStoremanGroupQuota(address tokenOrigAddr, address storemanGroup,
uint quota)
    external
    onlyRegistedOldSmgAdmin
    returns (bool)
    {
        return false;
    }

    /// @notice                                 gent function for oldStoremanGroupAdmin
to set storeman group's quota
    /// @param tokenOrigAddr                    token address of original chain
    /// @param storemanGroup                    storemanGroup address
    /// @param isNormal                         whether this unregister operation is in
normal condition
```
// 成都链安 // 完成 smg 注销(仅可由原 StoremanGroupAdmin 合约调用)
```
    function unregisterStoremanGroup(address tokenOrigAddr, address storemanGroup,
bool isNormal)
    external
    onlyRegistedOldSmgAdmin
    returns (bool)
    {
        //require(mapRegistedOldSmgAdmin[msg.sender], "It is not an valid old
storeman group admin object");
```
// 成都链安 // 调用 QuotaLedger 合约的 unregisterStoremanGroup 函数，完成注销 smg 操作
```
        bool ifSucc =
QuotaInterface(quotaLedger).unregisterStoremanGroup(tokenOrigAddr, storemanGroup,
isNormal);

        return ifSucc;
    }

}
```

// 成都链安 // ###### StoremanGroup 合约 ######
```
/*

  Copyright 2018 Wanchain Foundation.
  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
*/

//                            _           _           _
//  __      ____ _ _ __   ___| |__   __ _(_)_ __   __| | _____   __
//  \ \ /\ / / _` | '_ \ / __| '_ \ / _` | | '_ \@/ _` |/ _ \ \ / /
//   \ V  V / (_| | | | | (__| | | | (_| | | | | | (_| |  __/\ V /
//    \_/\_/ \__,_|_| |_|\___|_| |_|\__,_|_|_| |_|\__,_|\___| \_/
//
//  Code style according to: https://github.com/wanchain/wanchain-
token/blob/master/style-guide.rst

pragma solidity ^0.4.24; // 成都链安 // 建议固定编译器版本，并消除编译器告警

import "./SafeMath.sol";
import "./Halt.sol";

contract StoremanGroup is Halt {

    using SafeMath for uint; // 成都链安 // 引用 SafeMath 库，用于安全数学运算
    enum SCStatus
{Invalid,StakerElection,Lottery,Initial,Registered,Unregistered,Withdrawed,WorkDone}

    ///the minumum deposit for each depositor which is reuqired by storeman group
    uint public constant    DEFAULT_PRECISE     = 10000; // 成都链安 // 默认精度
    uint public constant    MIN_DEPOSIT         =  10000000000000000000000; // 成都链安
// 最小抵押额度
    uint public constant    MAX_COMMISION_BASE  =  50000000000000000000000; // 成都链安
// 参与抵押奖励的最大额度
    uint public constant    MAX_DEPOSIT_SUM     = 30000000000000000000000; // 成都链安
// 抵押上限

    ///the storeman group administration contract address
    address public      storemanGroupAdmin;
    address public      locatedMpcAddr; // 成都链安 // 合约用于注册 smg 的指定地址
    address public       locatedLotteryAddr;

    SCStatus public      scStatus = SCStatus.Invalid; // 成都链安 // 合约初始状态为
 "Invalid"

    ///the total lottery bonus
    uint public          totalLotteryBonus; // 成都链安 // 乐透奖励总额

    ///the distributed lottery bonus
    uint public          distributedLotterBonus; // 成都链安 // 已发送的乐透奖励金额

    ///the total total deposit of smg during staking phase
    uint public          totalStakingDeposit; // 成都链安 // 抵押总金额

    ///the final deposit of all stakers of smg
    uint public          totalSmgDeposit; // 成都链安 // 合约已确认的抵押总金额

    ///the total amount of the effective deposit that's used to divide bonus
    uint public          totalStakerQuota; // 成都链安 // 有效抵押额度(用于分配抵押奖励)

    ///the deposited quota recorded
    uint public          depositedQuota; // 成都链安 // 合约已注册 smg 的累计金额

    ///the number of supported token type
    uint public          totalTokenTypeNum; // 成都链安 // 合约注册 smg 的代币类型数量

    ///total bonus of smg from admin
    uint public          totalBonus; // 成都链安 // 系统抵押奖励总额(分发给已确认抵押用户)

    /// bonus commision per unit
    uint public          bonusPerUnit; // 成都链安 // 每份抵押金的抵押奖励额度

    ///total number of stakers
    uint public          stakerNum; // 成都链安 // 已确认的抵押用户数

    ///the flag which show whether the total deposit of smg stakers reach
MAX_DEPOSIT_SUM
    bool public          isReachedMaxDeposit = false; // 成都链安 // 标识是否已达抵押上
限，初始为 "false"

    ///storemanGroup running time
    uint public          runningStartTime; // 成都链安 // 活动开始时间
    uint public          runningEndTime; // 成都链安 // 活动结束时间
    ///staking  time
    uint public         stakingStartTime; // 成都链安 // 抵押开始时间
    uint public         stakingEndTime; // 成都链安 // 抵押结束时间
    // 成都链安 // 抵押信息结构体，存储抵押用户信息
    struct BidderInfo{
        uint    deposit;        //1 the deposit of this depositor
        uint    stakerRank;     //2 the rank of staker sequence
        uint    stakerQuota;    //3 the effective quota of deposit
        uint    stakerBonus;    //4 the bonus for staker
        uint    lotterRank;     //5 lotter rank
        uint    lotterBonus;    //6 lotter bonus
        bool    hasRevoked;     //7 if depositor has been revoked
        uint    bidTimeStamp;   //8 timestamp of the bidder's first deposit
    }
    // 成都链安 // 注册代币结构体，存储代币信息
    struct tokenDeposit{
        uint        deposit;        //1 the deposit of this depositor
        SCStatus    statu;          //2 the statu of tokenDeposit
        uint        revokeQuota;    //3 the quota revoked from token
    }

    ///the depositor info
    mapping(address => BidderInfo) public   mapDepositorInfo; // 成都链安 // 存储指定地
址的抵押信息

    ///the token supported (mapping: tokenOrigAddr -> tokenDeposit)
    mapping(address => tokenDeposit) public mapTokenSmgStatus; // 成都链安 // 存储指定代
币信息


    /**
    *
    * EVENTS
    *
    */
    /// @notice                         event for staking record
    /// @param _depositorAddr           depositor's address
    /// @param _depositValue            deposit
    /// @param _stakerAccumDeposit      accumulative deposit of depositor by now
    /// @param _smgAccumDeposit         accumulative deposit of smg by now
    /// @param _depositTimeStamp        accumulative deposit of smg by now
    event StoremanGroupDepositLogger(address indexed _depositorAddr, uint256
_depositValue, uint256 _depositTimeStamp, uint256 _stakerAccumDeposit, uint256
_smgAccumDeposit);

    /// @notice                         event for staker info
    /// @param _stakerAddr              the address of staker
    /// @param _depositValue             the deposit of staker
    /// @param _quota                    the effective quota of staker, must more 
than 1000 and no more than 5000
    /// @param _rank                     the rank of staker
    /// @param _accumStakerDeposit       the accumulative deposit of staker that 
has confirmed by now
    event StoremanGroupStakerLogger(address indexed _stakerAddr, uint256 
_depositValue, uint256 _quota, uint256 _rank, uint256 _accumStakerDeposit);

    /// @notice                          event for foundation to inject lottery 
bonus
    /// @param _foundationAddr           the address of foundation
    /// @param _bonus                    the bonus value injected for lottery
    /// @param _totalLotterBonus         total bonus for lottery
    /// @param _lotterNum                total number of lottery
    event StoremanGroupInjectLotteryBonusLogger(address indexed _foundationAddr, 
uint256 _bonus, uint256 _totalLotterBonus, uint256 _lotterNum);

    /// @notice                          event for foundation to inject lottery 
bonus
    /// @param _foundationAddr           the address of foundation
    /// @param _bonus                    the bonus value injected for smg
    /// @param _totalBonus               total bonus for smg by now
    event StoremanGroupInjectSmgBonusLogger(address indexed _foundationAddr, uint256 
_bonus, uint256 _totalBonus);

    /// @notice                          event for lotter info
    /// @param _lotterAddr               the address of lotter
    /// @param _lotterDeposit            the deposit of lotter
    /// @param _lotterBonus              the bonus of lotter
    /// @param _lotterRank               lottery rank
    event StoremanGroupLotteryLogger(address indexed _lotterAddr, uint256 
_lotterDeposit, uint256 _lotterBonus, uint256 _lotterRank);

    /// @notice                          event for smg to refeem  deposit from 
admin
    /// @param _depositorAddr            the address of depositor
    /// @param _deposit                  the deposit of depositor
    /// @param _bonus                    the lotter bonus of depsoit
    event DepositorRefeemAssertLogger(address indexed _depositorAddr, uint256 
_deposit, uint256 _bonus);

    /// @notice                          event for smg to refeem total bonus from 
admin
    /// @param _smgAmin                  the address of smg admin
    /// @param _tokenOrigAddr            token origin
    /// @param _smgRegAddr               the storeman group register address
    /// @param _originalChainAddr        the storeman group info on original chain
    /// @param _deposit                     total deposit that smg refeem from admin
    event StoremanGroupApplyRegisterLogger(address indexed _smgAmin, address indexed
_tokenOrigAddr, address indexed _smgRegAddr, address _originalChainAddr, uint256
_deposit, uint256 _txFeeRatio);

    /// @notice                             event for smg to refeem total bonus from
admin
    /// @param _smgAmin                     the address of smg
    /// @param _tokenOrigAddr               token origin address
    /// @param _smgRegAddr                  storemanGroup registed address
    event StoremanGroupApplyUnRegisterLogger(address indexed _smgAmin, address
indexed _tokenOrigAddr, address indexed _smgRegAddr);

    /// @notice                             event for smg to refeem  deposit from
admin
    /// @param _smgAmin                     smg address
    /// @param _tokenOrigAddr               token origin address
    /// @param _smgRegAddr                  the storeman group register address
    /// @param _depositRevoked              deposit that smg refeem from admin by one
token type
    /// @param _restDeposit                 total rest deposit that still deposited in
admin
    event StoremanGroupRefeemDepositLogger(address indexed _smgAmin, address indexed
_tokenOrigAddr, address indexed _smgRegAddr, uint256 _depositRevoked, uint256
_restDeposit);

    /// @notice                             event for staker to refeem  assert from
smg
    /// @param _stakerAddr                  the address of staker
    /// @param _deposit                     the deposit of staker
    /// @param _bonus                       staker's bonus deserved from smg
    event StakerRefeemAssetLogger(address indexed _stakerAddr, uint256 _deposit,
uint256 _bonus);

    /// @notice                             event of exchange WERC20 token with ERC20
token request
    /// @param _objectSmgAddr               address of storemanGroup where debt will
be transfered
    /// @param _tokenOrigAddr               address of ERC20 token
    /// @param _locatedMpcAddr              address of current smg locked Mpc node
    /// @param _debtValue                   debt value of which needed to transfer to
_objectSmgAddr
    event StoremanGroupDebtTransferLogger(address indexed _objectSmgAddr, address
indexed _tokenOrigAddr, address indexed _locatedMpcAddr, uint256 _debtValue);


    /**
     * MANIPULATIONS
     */
    /// @notice default transfer to contract
    // 成都链安 // 回退函数
    function () public payable {

        if((now >= stakingStartTime) &&(now <= stakingEndTime)){
            handleSmgStaking(msg.sender, msg.value); // 成都链安 // 如果处于抵押阶段，则
调用 handleSmgStaking 函数
        }else if(msg.sender == storemanGroupAdmin){ // 成都链安 // 否则，只允许
smgAdmin 向合约发送 WETH(smg 注销时退款)

        }else{
            revert("invalid assets inject!");
        }
    }

    /// @notice                           for setting smg admin
    /// @param _smgAdmin                  smg admin address
    /// @param _smgLottery                smg lottery sc address
    /// @param _smgRegAddr                the storeman group register address
    // 成都链安 // 合约初始配置(仅在 halted 为 true 时,可由 owner 调用)
    function InjectDependencies(address _smgAdmin,address _smgLottery,address
_smgRegAddr)
    public
    isHalted
    onlyOwner
    {
        // 成都链安 // 输入地址参数非零检查
        require((_smgAdmin != address(0)) && (_smgLottery != address(0)) &&
(_smgRegAddr != address(0)), "this is not valid smg admin");
        // 成都链安 // 要求本次操作在合约活动开始前，或者还未设置活动起止时间
        require((now < runningStartTime) || (0 == runningStartTime), "SmgAddr must
be set before running start");
        // 成都链安 // 设置合约关键变量
        storemanGroupAdmin = _smgAdmin;
        locatedLotteryAddr = _smgLottery;
        locatedMpcAddr = _smgRegAddr;
    }

    /// @notice                           for setting start and stop time for staking
phase
    /// @param _stakingStartTime          staking start time
    /// @param _stakingEndTime            staking end time
    // 成都链安 // 设置抵押阶段起止时间(仅在 halted 为 true 时,可由 owner 调用)
    function setSmgStakingTime(uint256 _stakingStartTime, uint256 _stakingEndTime)
    public
    isHalted
    onlyOwner
    {
        // 成都链安 // 要求当前时间戳小于抵押开始时间或者还未设置抵押时间
        require((0 == stakingStartTime) || (now < stakingStartTime), "Cannot config
staking time any more");
        // 成都链安 // 输入参数有效性检查
        require((_stakingStartTime > now) && (_stakingStartTime < _stakingEndTime),
"Invalid staking time");
        // 成都链安 // 设置抵押阶段起止时间
        stakingStartTime = _stakingStartTime;
        stakingEndTime = _stakingEndTime;
    }

    /// @notice                     for setting start and stop time for running
phase
    /// @param _runningStartTime         smg running start time
    // 成都链安 // 设置活动起止时间(仅在 halted 为 true 时，可由 owner 调用)
    function setSmgRunningTime(uint256 _runningStartTime, uint256 _runningEndTime)
    public
    isHalted
    onlyOwner
    {
        // 成都链安 // 要求在执行本次操作时，已设置有效抵押起止时间；并且当前时间戳小于活动开
始时间或者还未设置活动时间
        require((0 != stakingEndTime) && ((0 == runningStartTime) || (now <
runningStartTime)), "Cannot config staking time any more");
        // 成都链安 // 输入参数有效性检查，要求设置的活动开始时间大于当前时间戳和抵押结束时间
        require((_runningStartTime > now) && (_runningStartTime > stakingEndTime),
"Invalid running start time");
        // 成都链安 // 输入参数有效性检查，要求设置的活动结束时间大于当前时间戳和活动开始时间
        require((_runningEndTime > now) && (_runningStartTime < _runningEndTime),
"Invalid running end time");
        // 成都链安 // 设置活动起止时间
        runningStartTime = _runningStartTime;
        runningEndTime = _runningEndTime;
    }

    /// @notice                     to record smg depositor info
    /// @param _account              the depositor address
    /// @param _deposit              the deposit value
    // 成都链安 // 记录抵押信息(在 halted 为 false 时可调用)
    function handleSmgStaking(address _account, uint256 _deposit)
    private
    notHalted
    {
        require(_account != address(0), "Invalid account"); // 成都链安 // 抵押地址非零检查

        if(mapDepositorInfo[_account].deposit != 0){ // 成都链安 // 如果该用户已参与过抵押，则更新相关数据
            mapDepositorInfo[_account].deposit = mapDepositorInfo[_account].deposit.add(_deposit); // 成都链安 // 更新用户抵押总量

            totalStakingDeposit = totalStakingDeposit.add(_deposit); // 成都链安 // 更新系统抵押总金额
            // emit depositor info
            emit StoremanGroupDepositLogger(_account, _deposit, now, mapDepositorInfo[_account].deposit, totalStakingDeposit);
        }else{ // 成都链安 // 如果是新用户，则需要添加用户基本信息
            //construct mapDepositorInfo
            require(_deposit >= MIN_DEPOSIT, "Invalid deposit"); // 成都链安 // 检查抵押数量，不能低于最小抵押额度（仅在用户第一次抵押时检查)
            // 成都链安 // 设置用户信息
            mapDepositorInfo[_account].deposit = _deposit;
            // mapDepositorInfo[_account].stakerQuota = 0;
            // mapDepositorInfo[_account].stakerBonus = 0;
            // mapDepositorInfo[_account].stakerRank = 0;
            // mapDepositorInfo[_account].lotterRank = 0;
            // mapDepositorInfo[_account].lotterBonus = 0;
            mapDepositorInfo[_account].hasRevoked = false;
            mapDepositorInfo[_account].bidTimeStamp = now;

            totalStakingDeposit = totalStakingDeposit.add(_deposit); // 成都链安 // 更新系统的抵押总金额
            // emit depositor info
            emit StoremanGroupDepositLogger(_account, _deposit, now, _deposit, totalStakingDeposit);
        }
    }

    /// @notice                     to confirm final valid staker
    /// @param _accountArray        the address array of staker
    // 成都链安 // 确认抵押(在 halted 为 false 时，可由 owner 调用)
    function setSmgStakerInfo(address[] memory _accountArray)
    public
    onlyOwner
    notHalted
    {
        // 成都链安 // 要求抵押阶段已结束，且未达到抵押上限
        require((!isReachedMaxDeposit) && (now > stakingEndTime), "Invalid staker
setting operation");

        if(SCStatus.StakerElection != scStatus){
            scStatus = SCStatus.StakerElection; // 成都链安 // 修改合约状态为
 “StakerElection”
        }

        for(uint i = 0; i < _accountArray.length; i++){ // 成都链安 // 遍历抵押用户
            if(isReachedMaxDeposit){
                break; // 成都链安 // 如果已达到抵押上限，则结束遍历
            }

            address account = _accountArray[i];
            // 成都链安 // 抵押用户信息确认需满足：(1)抵押用户地址不为零地址；(2)用户抵押金额
不小于最小抵押额度；(3)该抵押用户未被确认
            require((address(0) != account) && (MIN_DEPOSIT <=
mapDepositorInfo[account].deposit), "Invalid staker info");
            require((0 == mapDepositorInfo[account].stakerRank) && (0 ==
mapDepositorInfo[account].stakerQuota), "Invalid staker info");

            uint finalDeposit = mapDepositorInfo[account].deposit; // 成都链安 // 获取该
用户最终抵押额度

            uint valideQuota;
            // 成都链安 // 获取该用户的最终有效抵押额度(有效抵押额度不能超过参与抵押奖励的最
大抵押额度)
            if(finalDeposit > MAX_COMMISION_BASE){
                valideQuota = MAX_COMMISION_BASE; // 成都链安 // 如果用户最终有效抵押数量
大于参与抵押奖励的最大抵押额度，则修改最终有效抵押数量为参与抵押奖励的最大抵押额度
            }else{
                valideQuota = finalDeposit;
            }

            stakerNum = stakerNum.add(1); // 成都链安 // 更新已确认的抵押用户数量
            mapDepositorInfo[account].stakerQuota = valideQuota; // 成都链安 // 设置用户
抵押的有效额度(用于计算抵押奖励)
            mapDepositorInfo[account].stakerRank = stakerNum; // 成都链安 // 设置用户的
抵押序号

            totalStakerQuota = totalStakerQuota.add(valideQuota); // 成都链安 // 更新合
约有效抵押总额
            totalSmgDeposit = totalSmgDeposit.add(finalDeposit); // 成都链安 // 更新合约
已确认的抵押总额

            if(totalSmgDeposit >= MAX_DEPOSIT_SUM){
                isReachedMaxDeposit = true; // 成都链安 // 修改已达抵押上限标识为 “true”
            }

            //emit final staker info
            emit StoremanGroupStakerLogger(account, finalDeposit, valideQuota,
stakerNum, totalSmgDeposit);
        }
    }

    /// @notice                            to inject lottery bonus
    /// @param _lotterNum                  the numbler of total lotter
    // 成都链安 // 注入乐透奖励资金(在 halted 为 false 时，可由 owner 调用)
    function injectLotteryBonus(uint256 _lotterNum)
    public
    payable
    notHalted
    onlyOwner
    {
        // 成都链安 // 调用者携带 WETH 数量和参数有效性检查，要求大于 0
        require((msg.value > 0) && (_lotterNum > 0), "Invalid lottery info, please
confirm lottery bonus");
        require(address(0) != locatedLotteryAddr, "Invalid lottery instance"); // 成
都链安 // 非零地址检查

        totalLotteryBonus = totalLotteryBonus.add(msg.value); // 成都链安 // 更新乐透奖
励总额

        bytes4 methodId = bytes4(keccak256("setSmgLotteryInfo(uint256,uint256)"));
// 成都链安 // 获取 setSmgLotteryInfo(uint256,uint256)函数选择器
        bool res = locatedLotteryAddr.call(methodId, totalLotteryBonus, _lotterNum);
// 成都链安 // 调用 StoremanLottery 合约的 setSmgLotteryInfo 函数，设置乐透数据
        require(res, "setSmgLotteryInfo failed"); // 成都链安 // 检查调用结果
        emit StoremanGroupInjectLotteryBonusLogger(msg.sender, msg.value,
totalLotteryBonus, _lotterNum);
    }

    /// @notice                            to inject smg bonus
    // 成都链安 // 注入抵押奖励资金(在 halted 为 false 时，可由 owner 被调用)
    function injectSmgBonus()
    public
    payable
    notHalted
    onlyOwner
    {
        require(msg.value > 0, "Invalid bonus"); // 成都链安 // 函数调用者携带的 WETH 数
量检查，要求大于 0
        totalBonus = totalBonus.add(msg.value); // 成都链安 // 更新抵押奖励总额
        bonusPerUnit = totalBonus.mul(DEFAULT_PRECISE).div(totalStakerQuota); // 成都
链安 // 计算每份抵押金的抵押奖励额度

        emit StoremanGroupInjectSmgBonusLogger(msg.sender, msg.value, totalBonus);
    }

    /// @notice                          to set lotter bonus for depositor
    /// @param _accountArray             the array of lotter address
    /// @param _bonus                    the bonus from lottery
    /// @param _rank                     the rank of bonus
    // 成都链安 // 向乐透中奖用户进行退款，并发送乐透奖励(仅在 halted 为 false 时，可由 owner
调用)
    function setSmgLotterInfo(address[] memory _accountArray, uint256 _bonus,
uint256 _rank)
    public
    onlyOwner
    notHalted
    {
        // 成都链安 // 该退款操作需满足：(1)当前时间戳大于抵押结束时间；(2)乐透奖金大于 0；(3)
抵押已达最大抵押上限
        require((now > stakingEndTime) && (totalLotteryBonus > 0) &&
isReachedMaxDeposit, "Not allowed to lotter just now");
        // 成都链安 // (4)剩余的乐透奖金足够支持本轮奖励发放
        require((totalLotteryBonus.sub(distributedLotterBonus)) >=
(_bonus.mul(_accountArray.length)), "There is no enough bonus for lottery");
        // 成都链安 // (5)输入的参数_rank 和_bonus 不为 0
        require((0 != _rank) && (0 != _bonus), "Invalid lotter bonus");

        for(uint i = 0; i < _accountArray.length; i++){ // 成都链安 // 遍历中奖用户
            address account = _accountArray[i];

            require(address(0) != account, "Invalid account"); // 成都链安 // 用户地址非
零检查
            // 成都链安 // 该退款操作需满足：(6)该用户参与了抵押，但未被 owner 所确认
            require((0 != mapDepositorInfo[account].deposit) && (0 ==
mapDepositorInfo[account].stakerRank), "Invalid lotter info");
```

```
            // 成都链安 // 该退款操作需满足：(7)该用户未领取乐透奖励
            require((0 == mapDepositorInfo[account].lotterBonus) && (0 ==
mapDepositorInfo[account].lotterRank), "Invalid lotter info");
            // 成都链安 // 更新该用户的乐透中奖信息
            mapDepositorInfo[account].lotterBonus = _bonus;
            mapDepositorInfo[account].lotterRank = _rank;
            // 成都链安 // 计算应退款的金额(抵押金额 + 奖励金额)
            uint lotterAsset =
mapDepositorInfo[account].deposit.add(mapDepositorInfo[account].lotterBonus);

            mapDepositorInfo[account].hasRevoked = true; // 成都链安 // 修改退款标识为
“true”
            distributedLotterBonus = distributedLotterBonus.add(_bonus); // 成都链安 //
更新已发送的乐透奖励金额

            //emit lottery info
            account.transfer(lotterAsset); // 成都链安 // 发送应退款金额至中奖用户
            emit StoremanGroupLotteryLogger(account,
mapDepositorInfo[account].deposit, _bonus, _rank);
        }

    }

    /// @notice                    to inject lottery bonus
    // 成都链安 // 完成乐透开奖(仅在 halted 为 false 时，可由 owner 调用)
    function finishLottery()
    public
    onlyOwner
    notHalted
    {
        scStatus = SCStatus.Lottery; // 成都链安 // 修改合约状态为“Lottery”
    }

    /// @notice                    to withdraw depositor's assert that deposit
and bonus
    /// @param _accountArray       the array of depositors address
    // 成都链安 // 向指定用户退款(仅在 halted 为 false 时，可由 owner 调用)
    function revokeDepositorsAsset(address[] memory _accountArray)
    public
    notHalted
    onlyOwner
    {
        require(scStatus == SCStatus.Lottery, "Cannot revoke non-staker's asset
now"); // 成都链安 // 要求合约状态为“Lottery”，表示该操作在发送乐透奖励之后
```

```
        for(uint i = 0; i < _accountArray.length; i++){ // 成都链安 // 遍历抵押用户
            address account = _accountArray[i];
            // 成都链安 // 抵押用户退款有效性检查:(1)用户地址不为零地址；(2)抵押资金不为 0
            require((address(0) != account) && (0 !=
mapDepositorInfo[account].deposit), "Invalid depositor info");
            // 成都链安 // (3)该抵押用户未被确认；(4)该用户未进行退款
            require((0 == mapDepositorInfo[account].stakerRank) && (false ==
mapDepositorInfo[account].hasRevoked), "Invalid depositor info");

            mapDepositorInfo[account].hasRevoked = true; // 成都链安 // 更新用户已退款标
识为“true”

            account.transfer(mapDepositorInfo[account].deposit); // 成都链安 // 发送抵押
资金至用户

            emit DepositorRefeemAssertLogger(account,
mapDepositorInfo[account].deposit, mapDepositorInfo[account].lotterBonus);
        }

    }

    /// @notice                    to inject lottery bonus
    // 成都链安 // 完成退款(仅在 halted 为 false 时，可由 owner 调用)
    function finishDepositorAssetRevoke()
    public
    onlyOwner
    notHalted
    {
        scStatus = SCStatus.Initial; // 成都链安 // 修改合约状态为“Initial”
    }

    /// @notice                    to withdraw depositor's assert that deposit
and bonus
    /// @param _stakerArray        the array of staker address
    // 成都链安 // 向已确认的抵押用户退款，并发送系统抵押奖励(仅在 halted 为 false 时，可由
owner 调用)
    function revokeStakersAsset(address[] memory _stakerArray)
    public
    notHalted
    onlyOwner
    {
        require(totalBonus > 0, "The Bonus from bund hasn't injected"); // 成都链安 //
要求当前的抵押奖励总额大于 0
        // 成都链安 // 要求当前时间戳大于活动结束时间，且合约状态为“Withdrawed”
        require((now > runningEndTime) && (scStatus == SCStatus.Withdrawed), "Cannot
revoke staker's asset now");

        for(uint i = 0; i < _stakerArray.length; i++){ // 成都链安 // 遍历抵押用户
            address account = _stakerArray[i];
            // 成都链安 // 抵押用户有效性检查:(1)用户地址不为零地址；(2)抵押额度不为 0
            require((address(0) != account) && (0 !=
mapDepositorInfo[account].deposit), "Invalid acount address");
            // 成都链安 // (3)该用户已被确认；(4) 未对该用户抵押资金进行退款
            require((0 != mapDepositorInfo[account].stakerRank) && (false ==
mapDepositorInfo[account].hasRevoked), "Invalid staker info");

            // to count the commision of this staker
            // 成都链安 // 计算抵押用户的抵押奖励
            mapDepositorInfo[account].stakerBonus =
(mapDepositorInfo[account].stakerQuota).mul(bonusPerUnit).div(DEFAULT_PRECISE);
            // 成都链安 // 计算应退款金额(抵押金额 + 奖励金额)
            uint stakerAsset =
mapDepositorInfo[account].deposit.add(mapDepositorInfo[account].stakerBonus);
            mapDepositorInfo[account].hasRevoked = true; // 成都链安 // 更新用户已退款标
识为 “true”

            // to update the record of totalSmgDeposit, totalBonus
            // 成都链安 // 检查合约已确认的抵押金额是否足够支持本次退款
            require(totalSmgDeposit >= (mapDepositorInfo[account].deposit), "There
is not enough deposit left for deposit revoking");
            // 成都链安 // 更新合约已确认的抵押总额
            totalSmgDeposit =
totalSmgDeposit.sub(mapDepositorInfo[account].deposit);
            // 成都链安 // 检查当前系统抵押奖励总额是否足够支持本次的奖励发放
            require(totalBonus >= (mapDepositorInfo[account].stakerBonus), "There is
not enough bonus left for bonus dispatch");
            // 成都链安 // 更新抵押奖励总额
            totalBonus = totalBonus.sub(mapDepositorInfo[account].stakerBonus);
            // 成都链安 // 发送应退款金额至用户
            account.transfer(stakerAsset);

            emit StakerRefeemAssetLogger(account, mapDepositorInfo[account].deposit,
mapDepositorInfo[account].stakerBonus);
        }
    }

    /// @notice                    to withdraw depositor's assert that deposit
and bonus
    // 成都链安 // 完成已确认的抵押用户退款(仅在 halted 为 false 时，可由 owner 调用)
    function finishStakersAssetRevoke()
    public
    onlyOwner
    notHalted
    {
        scStatus = SCStatus.WorkDone; // 成都链安 // 修改合约状态为“WorkDone”
    }

    /// @notice                     function for storeman registration, this
method should be invoked by the storemanGroup himself
    /// @param _tokenOrigAddr       token address of original chain
    /// @param _originalChainAddr   the storeman group info on original chain
    /// @param _txFeeRatio          the transaction fee required by storeman
group
    // 成都链安 // 注册 smg(仅在 halted 为 false 时，可由 owner 调用)
    function applyRegisterSmgToAdmin(address _tokenOrigAddr, address
_originalChainAddr, uint256 _txFeeRatio, uint256 _deposit)
    public
    onlyOwner
    notHalted
    {
        // 成都链安 // 输入地址参数非零检查
        require((address(0) != locatedMpcAddr) && (_tokenOrigAddr != address(0)) &&
(_originalChainAddr != address(0)), "Invalid token address");
        // 成都链安 // 时间戳检查，要求当前时间戳大于抵押结束时间；并且当前时间戳小于活动结束
时间，或者未设置活动结束时间
        require((now > stakingEndTime) && ((now < runningEndTime) || (0 ==
runningEndTime)), "cannot regist smg to admin now");
        // 成都链安 // 要求合约状态为“Initial”或者“Registered”
        require((SCStatus.Initial == scStatus) || (SCStatus.Registered == scStatus),
"the smg statu cannot support regist now");
        // 成都链安 // 要求合约余额不小于输入的 smg 注册资金额度
        require(this.balance >= _deposit, "Invalid smg balance");
        // 成都链安 // 检查合约剩余的已确认抵押金额是否足够支持本次 smg 注册
        require(totalSmgDeposit >= depositedQuota.add(_deposit), "not enough depoist
for register smg to admin");

        if(SCStatus.Withdrawed == mapTokenSmgStatus[_tokenOrigAddr].statu){
            // 成都链安 // 如果注册的代币处于“Withdrawed”状态，则需检查代币的总资产量是
否等于退款数量
            require(mapTokenSmgStatus[_tokenOrigAddr].deposit ==
mapTokenSmgStatus[_tokenOrigAddr].revokeQuota, "this token has taken an inaccuracy
deposit revoke before ");
        }else{ // 成都链安 // 否则，要求注册的代币处于“Invalid”状态
            require(SCStatus.Invalid == mapTokenSmgStatus[_tokenOrigAddr].statu,
"this token type is listed by the smg by now");
        }
        // 成都链安 // 获取
storemanGroupRegisterByDelegate(address,address,address,uint256)的函数选择器
        bytes4 methodId =
bytes4(keccak256("storemanGroupRegisterByDelegate(address,address,address,uint256)")
);
        // 成都链安 // 调用 ImprovedStoremanGroupAdmin 合约
storemanGroupRegisterByDelegate 函数进行 smg 代理注册，其中代理人是本合约，smg 地址是
 'locatedMpcAddr'
        bool res = storemanGroupAdmin.call.value(_deposit)(methodId,
_tokenOrigAddr,locatedMpcAddr,_originalChainAddr,_txFeeRatio);
        require(res, "storemanGroupRegister failed"); // 成都链安 // 检查调用结果

        depositedQuota = depositedQuota.add(_deposit); // 成都链安 // 更新合约已注册 smg
的累计金额
        totalTokenTypeNum = totalTokenTypeNum.add(1); // 成都链安 // 更新合约注册 smg 的
代币类型数量
        // 成都链安 // 更新代币信息
        mapTokenSmgStatus[_tokenOrigAddr].deposit = _deposit;
        mapTokenSmgStatus[_tokenOrigAddr].statu = SCStatus.Registered;
        mapTokenSmgStatus[_tokenOrigAddr].revokeQuota = 0;

        scStatus = SCStatus.Registered; // 成都链安 // 更新合约状态为 "Registered"

        emit StoremanGroupApplyRegisterLogger(storemanGroupAdmin, _tokenOrigAddr,
locatedMpcAddr, _originalChainAddr, _deposit, _txFeeRatio);
    }

    /// @notice                           function for storemanGroup applying
unregister (needed to confirm further)
    /// @param _objectSmgAddr             address of smg which used to accept debt
    /// @param _value                     debt value that needed to transfer
    // 成都链安 // 检查是否可注销 smg(仅在 halted 为 false 时，可由 owner 调用)
    function applySmgDebtTransfer(address _objectSmgAddr, address _tokenOrigAddr,
uint256 _value)
    public
    notHalted
    onlyOwner
    {
        // 成都链安 // 非零地址检查
        require((address(0) != locatedMpcAddr)&&(address(0) !=
_objectSmgAddr)&&(address(0) != _tokenOrigAddr), "Invalid token original chain
address");
```

```
        require(_value > 0, "Invalid token original chain address"); // 成都链安 // 输
入参数有效性检查，要求大于 0
        // 成都链安 // 要求当前时间戳大于活动结束时间；并且对应代币的状态为“Registered”
        require((now > runningEndTime) && (scStatus == SCStatus.Registered), "cannot
unregister smg now");

        emit StoremanGroupDebtTransferLogger(_objectSmgAddr, _tokenOrigAddr,
locatedMpcAddr, _value);
    }

    /// @notice                           function for storemanGroup applying
unregister
    /// @param _tokenOrigAddr             token address of original chain
    // 成都链安 // 申请注销 smg(仅在 halted 为 false 时，可由 owner 调用)
    function applyUnregisterSmgFromAdmin(address _tokenOrigAddr)
    public
    notHalted
    onlyOwner
    {
        // 成都链安 // 非零地址检查
        require((address(0) != locatedMpcAddr) && (address(0) != _tokenOrigAddr),
"Invalid token original chain address");
        // 成都链安 // 要求当前时间戳大于活动结束时间；并且对应的代币状态为“Registered”
        require((now > runningEndTime) && (SCStatus.Registered ==
mapTokenSmgStatus[_tokenOrigAddr].statu), "cannot unregister smg now");
        // 成都链安 // 获取 smgApplyUnregisterByDelegate(address,address)的函数选择器
        bytes4 methodId =
bytes4(keccak256("smgApplyUnregisterByDelegate(address,address)"));
        // 成都链安 // 调用 ImprovedStoremanGroupAdmin 合约
smgApplyUnregisterByDelegate 函数进行 smg 代理注销
        bool res = storemanGroupAdmin.call(methodId, _tokenOrigAddr,
locatedMpcAddr);
        require(res, "smgApplyUnregisterByDelegate failed"); // 成都链安 // 检查调用结果
        // 成都链安 // 修改对应代币的状态为“Unregistered”
        mapTokenSmgStatus[_tokenOrigAddr].statu = SCStatus.Unregistered;

        emit StoremanGroupApplyUnRegisterLogger(storemanGroupAdmin, _tokenOrigAddr,
locatedMpcAddr);
    }


    /// @notice                          to withdraw storeman group's deposit
    /// @param _tokenOrigAddr            token address of original chain
    // 成都链安 // 合约所注册的 smg 退款(仅在 halted 为 false 时，可由 owner 调用)
    function withDrawSmgFromAdmin(address _tokenOrigAddr)
    public
    notHalted
    onlyOwner
    {
        // 成都链安 // 非零地址检查
        require((address(0) != locatedMpcAddr) && (address(0) != _tokenOrigAddr),
"Invalid token original chain address");
        // 成都链安 // 时间戳检查，要求当前时间戳大于活动结束时间；并且对应的代币状态为
“Unregistered”
        require((now > runningEndTime) && (SCStatus.Unregistered ==
mapTokenSmgStatus[_tokenOrigAddr].statu), "cannot withdraw smg now");
        uint preBalance = this.balance; // 成都链安 // 获取退款前合约余额
        // 成都链安 // 获取 smgWithdrawDepositByDelegate(address,address)的函数选择器
        bytes4 methodId =
bytes4(keccak256("smgWithdrawDepositByDelegate(address,address)"));
        // 成都链安 // 调用 ImprovedStoremanGroupAdmin 合约
smgWithdrawDepositByDelegate 函数进行对合约注册的 smg 进行退款
        bool res = storemanGroupAdmin.call(methodId, _tokenOrigAddr,
locatedMpcAddr);
        require(res, "smgWithdrawDepositByDelegate failed");

        uint aftBalance = this.balance; // 成都链安 // 获取退款后合约余额
        uint revokedDeposit = aftBalance.sub(preBalance); // 成都链安 // 获取本次退款金
额

        depositedQuota = depositedQuota.sub(revokedDeposit); // 成都链安 // 更新合约已注
册 smg 的累计金额
        totalTokenTypeNum = totalTokenTypeNum.sub(1); // 成都链安 // 更新合约注册 smg 的
代币类型数量

        mapTokenSmgStatus[_tokenOrigAddr].statu = SCStatus.Withdrawed; // 成都链安 //
更新 smg 对应代币的状态为“Withdrawed”
        mapTokenSmgStatus[_tokenOrigAddr].revokeQuota = revokedDeposit; // 成都链安 //
更新 smg 对应代币的退款数量

        if(0 == totalTokenTypeNum){
            scStatus = SCStatus.Withdrawed; // 成都链安 // 如果合约注册 smg 的代币类型数量
为 0，则修改合约状态为“Withdrawed”
        }

        emit StoremanGroupRefeemDepositLogger(storemanGroupAdmin, _tokenOrigAddr,
locatedMpcAddr, revokedDeposit, depositedQuota);
    }
    /// @notice function for destroy contract
    // 成都链安 // 自毁函数(仅在 halted 为 false 时，可由 owner 调用)
    function kill()
    public
    isHalted
    onlyOwner
    {
        require(scStatus == SCStatus.WorkDone, "cannot destruct smg now"); // 成都链安
// 要求合约状态为 “WorkDone”
        selfdestruct(owner); // 成都链安 // 销毁合约，并发送合约余额至 owner
    }

}

// 成都链安 // ###### StoremanLottery 合约 ######


/*

  Copyright 2018 Wanchain Foundation.

  Licensed under the Apache License, Version 2.0 (the "License");
  you may not use this file except in compliance with the License.
  You may obtain a copy of the License at

  http://www.apache.org/licenses/LICENSE-2.0

  Unless required by applicable law or agreed to in writing, software
  distributed under the License is distributed on an "AS IS" BASIS,
  WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
  See the License for the specific language governing permissions and
  limitations under the License.
*/

//                            _            _            _
//  __      ____ _ _ __   ___| |__   __ _(_)_ __   __| | _____   __
//  \ \ /\ / / _` | '_ \ / __| '_ \ / _` | | '_ \@/ _` |/ _ \ \ / /
//   \ V  V / (_| | | | | (__| | | | (_| | | | | | (_| |  __/\ V /
//    \_/\_/ \__,_|_| |_|\___|_| |_|\__,_|_|_| |_|\__,_|\___| \_/
//
//  Code style according to: https://github.com/wanchain/wanchain-
token/blob/master/style-guide.rst

pragma solidity ^0.4.24; // 成都链安 // 建议固定编译器版本

import "./SafeMath.sol";
import "./Halt.sol";

contract StoremanLottery is Halt{

    using SafeMath for uint; // 成都链安 // 引用 SafeMath 库，用于安全数学运算

    ///the hash of block that's created at stakingEndTime
    bytes32 public stakingEndBlockhash;

    ///the address of storeman group sc
    address public storemanGroupAddr;

    ///the random seed generated based on stakingEndBlockhash and other info
    uint256 public storemanRandomSeed; // 成都链安 // 随机数种子(依赖于最后一次抵押区块哈希
和 owner 地址)

    ///the total lottery bonus
    uint public totalLotteryBonus; // 成都链安 // 乐透奖励总额

    ///the total lotter numbers
    uint public totalLotterNum; // 成都链安 // 乐透总抽奖次数

    ///the sequence of lotter random that has been generated
    uint public recoredNum; // 成都链安 // 已抽奖次数

    ///the lottery bonus that has been distributed
    uint public recoredBonus; // 成都链安 // 已抽奖的奖金

    ///the array of random that has been generated
    uint[] public seededRandomArray; // 成都链安 // 随机数数组，存储产生的随机数

    ///the struct to record lottery rank, just include bonus and number for this
rank
    // 成都链安 // 乐透信息结构体，存储乐透抽奖数据(乐透奖励金额和乐透抽奖次数)
    struct LotterBonuInfo{
        uint lotterBonus;
        uint lotterNum;
    }

    ///the lotter info map: map(randomHash = > lotterBonusInfo)
    mapping(uint => LotterBonuInfo) public    mapSmgLotteryBonus;

    /// @notice                           event for smg to refeem total bonus from
admin
    /// @param _smgAddr                   the address of smg
    /// @param _seed                      the seed for random
    event LotteryRandomSeedLogger(address indexed _smgAddr, uint256 _seed);

    /// @notice                             event for smg to refeem total bonus from
admin
    /// @param _smgAddr                     the address of smg
    /// @param _rank                        the rank for lottery
    /// @param _random                      the seeded random
    /// @param _bonus                       the bonus for this lottery rank
    event LotterySeededRandomLogger(address indexed _smgAddr, uint256 _rank, uint256
_random, uint256 _bonus);

    /// @dev `owner` is the only address that can call a function with this
    /// modifier
    modifier onlyStoremanGroup() {
        require(msg.sender == storemanGroupAddr); // 成都链安 // 要求被修饰函数的调用者必
须是 StoremanGroup 合约
        _;
    }

    /// @notice                           to set storemangroup address
    /// @param _smgAddr                   the address of storeman group
    // 成都链安 // 设置 StoremanGroup 合约地址(仅由 owner 可调用)
    function setStoremanGroupAddr(address _smgAddr)
    public
    onlyOwner
    {
        // 成都链安 // 非零地址检查
        require(address(0) != _smgAddr, "Invalid storeman group object");
        storemanGroupAddr = _smgAddr; // 成都链安 // 设置 StoremanGroup 合约地址
    }

    /// @notice                           to set block hash when staking end time
    /// @param _bonus                     the bonus value of this lotter rank
    /// @param _num                       the lotter number of this lotter rank
    // 成都链安 // 设置乐透数据(仅可由 StoremanGroup 合约地址调用)
    function setSmgLotteryInfo(uint256 _bonus, uint256 _num)
    external
    onlyStoremanGroup
    {
        require((0 != _bonus) && (0 != _num), "Invalid lottery"); // 成都链安 // 参数非
零检查

        totalLotteryBonus = _bonus; // 成都链安 // 设置乐透奖励总额
        totalLotterNum = _num; // 成都链安 // 设置乐透总抽奖次数
    }
    /// @notice                       to get random seed for lottery
    // 成都链安 // 生成乐透抽奖的随机数种子
    function genLotteryRandomSeed(bytes32 _blockHash)
    public
    onlyOwner
    {
        // 成都链安 // 检查输入参数 ‘_blockHash’ 不为空
        require(bytes32(0) != _blockHash, " Invalid stakingEndBlockhash");

        stakingEndBlockhash = _blockHash; // 成都链安 // 设置 stakingEndBlockhash
        // 成都链安 // 生成乐透抽奖随机数种子
        storemanRandomSeed = uint256(keccak256(abi.encodePacked(stakingEndBlockhash,
msg.sender)));
        emit LotteryRandomSeedLogger(storemanGroupAddr, storemanRandomSeed);
    }

    /// @notice                       to get random seed for lottery
    /// @param _rank                  the rank of  lotter
    /// @param _num                   the number of this lotter rank
    /// @param _bonus                 the bonus of this lotter rank
    // 成都链安 // 生成乐透抽奖的随机数(仅可由 owner 调用)
    function genSeededRandom(uint256 _rank, uint256 _num, uint256 _bonus)
    public
    onlyOwner
    {
        // 成都链安 // 检查乐透抽奖的随机数种子不为 0
        require(uint256(0) != storemanRandomSeed, "storemanRandomSeed hasn't
generated");
        // 成都链安 // 要求相同等级不能重复生成随机数
        require(0 == mapSmgLotteryBonus[_rank].lotterNum, "This lotter rank has been
generated");

        uint restBonus;
        restBonus = totalLotteryBonus.sub(recoredBonus).sub(_num.mul(_bonus)); // 成
都链安 // 获取本轮乐透开奖后的剩余奖金
        require(restBonus >= 0, "The lottery bonus is not enough"); // 成都链安 // 要求
剩余奖金不小于 0
        require(totalLotterNum >= recoredNum.add(_num), "exceed the max lotters
number"); // 成都链安 // 要求本轮乐透的抽奖次数与已抽奖次数之和不能大于既定的总抽奖次数
        for(uint i = 0; i < _num; i++ ){
            recoredNum = recoredNum.add(1); // 成都链安 // 更新已抽奖次数
            // 成都链安 // 生成本次抽奖的随机数
            uint random = uint(keccak256(abi.encodePacked(storemanRandomSeed,
recoredNum, _bonus)));
            emit LotterySeededRandomLogger(storemanGroupAddr, _rank, random,
_bonus);

            seededRandomArray.push(random); // 成都链安 // 将生成的随机数加入指定数组进行
存储
            recoredBonus = recoredBonus.add(_bonus); // 成都链安 // 更新已抽奖金额
        }

        setSmgLotterRule(_rank, _bonus, _num); // 成都链安 // 调用 setSmgLotterRule 函数
记录该等级下的乐透抽奖数据
    }

    /// @notice                         to set block hash when staking end time
    /// @param _rank                    lotter rank
    /// @param _bonus                   the bonus value of this lotter rank
    /// @param _num                     the lotter number of this lotter rank
    // 成都链安 // 设置指定等级的乐透抽奖数据(内部函数)
    function setSmgLotterRule(uint256 _rank, uint256 _bonus, uint256 _num)
    private
    {
        // 成都链安 // 设置该等级下的乐透抽奖数据
        mapSmgLotteryBonus[_rank].lotterBonus = _bonus;
        mapSmgLotteryBonus[_rank].lotterNum = _num;
    }
    // 成都链安 // 自毁函数(在 halted 为 true 时，可由 owner 调用)
    function kill()
    public
    isHalted
    onlyOwner
    {
        selfdestruct(owner); // 成都链安 // 销毁合约，并发送合约余额至 owner
    }

    /// @notice If token coin is sent to this address, send it back.
    /// @dev If token coin is sent to this address, send it back.
    function ()
    public
    payable
    {
        revert();
    }

}
```

成都链安
BEOSIN

官方网址

https：//lianantech.com

电子邮箱

vaas@lianantech.com

微信公众号