Documentation for Trashcoin's Smart Contract

Overview
The `RecyclingIncentiveSystem` smart contract incentivizes recycling by rewarding users with vouchers for depositing recyclable materials. It involves multiple stakeholders: admins, vendors, and recycling personnel, each with specific roles in the system.

Key Features
1. **Trash Reward System**:
   - Users deposit recyclable materials and receive vouchers based on predefined reward rates for different material types.
2. **Administered Payouts**:
   - Admins manage payouts to investors in Ether or vouchers.
3. **Stakeholder Roles**:
   - Roles for admins, vendors, and recycling personnel are managed by the system to ensure secure operations.
4. **Configurable Rates**:
   - Admins can set reward rates for different types of materials.

Contract Details

State Variables
- **Stakeholders**:
  - address public adminPool: Stores the address of the admin voucher pool.
  - mapping(address => uint256) public vouchers: Tracks voucher balances for all addresses.
  - mapping(address => bool) public isAdmin: Indicates if an address is an admin.
  - mapping(address => bool) public isVendor: Indicates if an address is a vendor.
  - mapping(address => bool) public isRecyclingPersonnel: Indicates if an address belongs to recycling personnel.

- **Reward Configuration**:
  - enum MaterialType: Represents recyclable material types (Plastic, Glass, Metal, Paper).
  - mapping(MaterialType => uint256) public rewardRates: Stores reward rates for each material type.

Events
- TrashDeposited: Triggered when a user deposits trash and receives vouchers.
- TrashProcessed: Triggered when recycling personnel process trash.
- VendorDeposit: Triggered when a vendor deposits vouchers into the admin pool.
- PayoutWithdrawn: Triggered when an admin withdraws payouts to an investor.
- AdminAdded: Triggered when a new admin is added.
- VendorAdded: Triggered when a new vendor is added.
- RecyclingPersonnelAdded: Triggered when recycling personnel are added.
- VouchersAdded: Triggered when vouchers are added to an account.
- VendorCheck: Triggered when a vendor check is performed.

Modifiers

- onlyAdmin: Ensures only admin addresses can call the function.
- onlyVendor: Ensures only vendor addresses can call the function.
- onlyRecyclingPersonnel: Ensures only recycling personnel can call the function.

Constructor
- Initializes the contract with:
  - The admin pool address.
  - The contract deployer as the first admin.
  - Default reward rates for each material type.

Functions

Stakeholder Management(This Project Does Not Aim To Promote Stakeholder Capitalism)
1. addAdmin(address _admin):
   - Adds a new admin address.
   - Emits "AdminAdded".

2. addVendor(address _vendor):
   - Adds a new vendor address.
   - Emits "VendorAdded".

3. addRecyclingPersonnel(address _personnel):
   - Adds a new recycling personnel address.
   - Emits "RecyclingPersonnelAdded".

Trash Management
1. depositTrash(MaterialType material, uint256 trashWeight):
   - Calculates rewards based on the material type and weight.
   - Deducts vouchers from the admin pool and credits the user.
   - Emits "TrashDeposited".

2. processTrash(uint256 trashValue):
   - Processes trash and deducts the corresponding voucher value from the admin pool.
   - Emits "TrashProcessed".

3. setRewardRate(MaterialType material, uint256 rate):
   - Allows admins to set reward rates for specific material types.

Vendor Operations
1. vendorDeposit(uint256 voucherAmount):
   - Allows vendors to deposit vouchers into the admin pool.
   - Emits "VendorDeposit".

Payouts
1. withdrawPayout(address payable investor, uint256 ethAmount, uint256 voucherAmount):
   - Admins can withdraw Ether or vouchers for an investor.
   - Transfers Ether directly and updates voucher balances.
   - Emits "PayoutWithdrawn".

Utility
1. balanceOf(address account):

   - Returns the voucher balance for an address.

2. receive():
   - Allows the contract to receive Ether directly.

Security and Constraints

1. Role-based access control (onlyAdmin, onlyVendor, onlyRecyclingPersonnel) ensures only authorized users can perform sensitive operations.
2. Overflow checks and zero-address validation protect against invalid transactions.
3. Ether transfers use safe call methods to prevent failure.

Usage Examples

Adding a New Admin

```
contract.addAdmin(newAdminAddress);
```

Depositing Trash

```
contract.depositTrash(MaterialType.Plastic, 10); // 10 kg of Plastic
```

Setting Reward Rate

```
contract.setRewardRate(MaterialType.Glass, 2.5 ether); // Update Glass reward rate to 2.5
vouchers/kg
```

Withdrawing Payouts

```
contract.withdrawPayout(payable(investorAddress), 1 ether, 50); // 1 ETH + 50 vouchers
```