**SBP-DID-Solidity合约设计说明书V0.1**

1. **编写目的**

为了让项目组成员对SBP-DID合约的整体设计有一个全面详细的了解，同时为项目的开发、测试、验证、交付等环节提供原始依据以及开发指导，特此整理SBP-DID合约整体设计规范方案说明文档。

2. **设计规范**

2.1 **安全性设计说明**

SBP-DID合约的整体设计目前采用2层设计模式，分别是代理合约和业务合约，业务合约只允许与之对应的代理合约进行调用。

注：业务合约都需要定义相应的接口，业务处理在业务合约中进行实现。

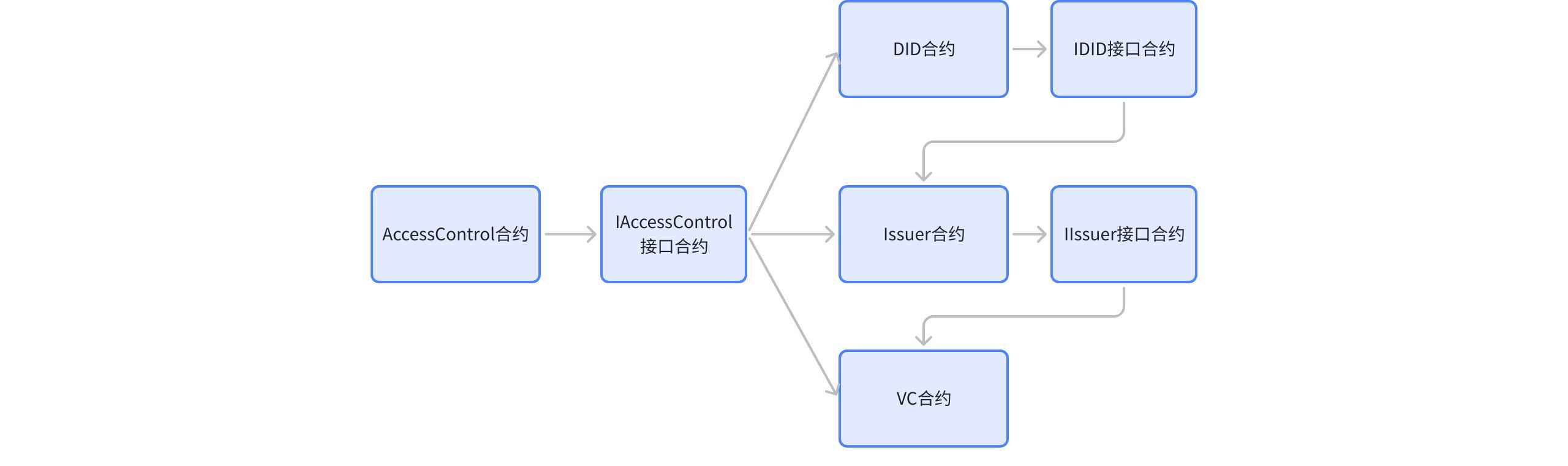
2.2 **合约更新设计说明**

SBP-DID合约通过UUPS（EIP-1822: Universal Upgradeable Proxy Standard）模式实现其业务合约的可升级。SBP-DID的各个合约均有一个代理合约。

2.2.1 **业务合约的编写**

1. 继承UUPSUpgradeable。该类库实现了UUPS代理设计的可升级机制。
2. 修改初始化方法initialize()。代理合约部署时需要调用以进行合约的初始化操作。
3. AccessControl合约对外提供对应IAccessControl接口合约，供DID合约、Issuer合约、VC合约调用

DID合约对外提供对应IDID接口合约，供Issuer合约调用。Issuer合约对外提供对应IIssuer接口合约，供VC合约调用。



2.2.2 **业务合约的部署过程**

1. 部署业务合约。
2. 部署代理合约。部署时构造传参写入业务合约地址、initialize的方法签名，实现其与业务合约的映射以及初始化操作。

2.2.3 **业务合约的升级过程**

1. 部署新版本的业务合约。
2. 调用当前代理合约中的upgradeTo方法。执行时传入新的业务合约地址，实现其与新版本业务合约的映射，达到升级的目的。

3. **合约设计**

3.1 **AccessControl合约**

3.1.1 **功能介绍**

基于函数选择器（bytes4）提供外部节点要加入的链账户地址权限控制，其中每个链账户地址可以具有不同的函数调用权限。支持批量授权和撤销（默认只有超级管理员（项目托管链账户）可进行设置），支持查询当前有效权限和根据单个链账户获取所有函数选择器；

包含对项目是否私有以及是否开启VC模版验证以及是否开启issuer验证的查询和更改方法；

合约本身控制权限基于OpenZeppelin AccessControlDefaultAdminRulesUpgradeable实现，使用超级管理员进行合约的初始化，并具有合约启用和停止控制功能。

生成函数选择器工具页面：https://chaintool.tech/querySelector

3.1.2 **数据结构**

* 项目是否私有

|  |
| --- |
| Solidity bool private \_isProjectPrivate |

* 项目是否启用VC模版验证

|  |
| --- |
| Solidity bool private \_enableVCTemplateVerification |

* 项目是否启用发证方验证

|  |
| --- |
| Solidity bool private \_enableIssuerVerification; |

* 项目did标识符里的方法名

|  |
| --- |
| Solidity string[] private \_method; |

* 链账户下函数选择器权限映射

|  |
| --- |
| Solidity mapping(address => mapping(bytes4 => bool)) private \_selectorPermissions; |

* 链账户下所有函数选择器映射

|  |
| --- |
| Solidity mapping(address => bytes4[]) private \_userSelectors; |

* 链账户下函数选择器对应\_userSelectors索引值映射（默认+1，0值代表不存在）

|  |
| --- |
| Solidity mapping(address => mapping(bytes4 => uint256)) private \_selectorIndex; |

* 链账户函数选择器列表结构体

|  |
| --- |
| Solidity struct AccountSelector {  address account;  bytes4[] selectors; } |

3.1.3 **方法定义**

3.1.3.1 **初始化**

合约初始化

* 方法：

|  |
| --- |
| Solidity function initialize(  string memory method,  bool isPrivate,  bool enableVcVerification,  bool enableIssuerVerification ) public initializer |

* 入参：项目method名称，项目是否私有，是否开启VC模版审核，是否开启发证方审核
* 出参：N/A
* 事件：N/A
* 核心逻辑：

1. 初始化写入项目method名称，项目是否私有，是否开启VC模版审核，是否开启发证方审核

3.1.3.2 **更改项目开放状态**

更改项目开放或私有状态

* 方法：

|  |
| --- |
| Solidity function changePrivateStatus(bool isPrivate) public |

* 入参：是否私有
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity event PrivateStatusChanged(bool indexed isPrivate) |

* 核心逻辑：

1. 校验调用者是否为超级管理员，校验项目是否停用
2. 写入项目开放状态
3. 触发事件

3.1.3.3 **更改项目method**

更改项目method名称

* 方法：

|  |
| --- |
| Solidity function changeMethod(string memory method) public |

* 入参：项目method名称
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity event MethodChanged(string indexed method) |

* 核心逻辑：

1. 校验调用者是否为超级管理员，校验项目是否停用
2. 校验入参是否非空字符串，为空抛出异常并回滚交易
3. 更改项目method
4. 触发事件

3.1.3.4 **更改VC模版审核状态**

更改项目VC模版验证启用状态

* 方法：

|  |
| --- |
| Solidity function changeEnableVCTemplateVerification(bool enableVCTemplateVerification) public |

* 入参：是否开启
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity event EnableVCTemplateVerificationChanged(bool indexed enableVCTemplateVerification) |

* 核心逻辑：

1. 校验调用者是否为超级管理员，校验项目是否停用
2. 更改项目VC验证启用状态
3. 触发事件

3.1.3.5 **更改Issuer审核状态**

更改项目Issuer验证启用状态

* 方法：

|  |
| --- |
| Solidity function changeEnableIssuerVerification(bool enableIssuerVerification) public |

* 入参：是否开启
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity event EnableIssuerVerificationChanged(bool indexed enableIssuerVerification) |

* 核心逻辑：

1. 校验调用者是否为超级管理员，校验项目是否停用
2. 更改项目Issuer验证启用状态
3. 触发事件

3.1.3.6 **查询项目是否私有**

查询项目是否私有，私有返回true，公开返回false

* 方法：

|  |
| --- |
| Solidity function isProjectPrivate() public view returns (bool) |

* 入参：N/A
* 出参：私有返回true，公开返回false
* 事件：N/A
* 核心逻辑：

1. 校验项目是否停用
2. 返回结果

3.1.3.7 **查询项目method**

查询项目当前的method

* 方法：

|  |
| --- |
| Solidity function getMethod() public view returns (string memory) |

* 入参：N/A
* 出参：did
* 事件：N/A
* 核心逻辑：

1. 校验项目是否停用
2. 返回结果

3.1.3.8 **查询项目是否开启Issuer审核**

查询项目是否开启Issuer验证，开启返回true，关闭返回false

* 方法：

|  |
| --- |
| Solidity function isIssuerVerificationEnabled() public view returns (bool) |

* 入参：N/A
* 出参：开启返回true，关闭返回false
* 事件：N/A
* 核心逻辑：

1. 校验项目是否停用
2. 返回结果

3.1.3.9 **查询项目是否开启VC模版审核**

查询项目是否开启VC模版验证，开启返回true，关闭返回false

* 方法：

|  |
| --- |
| Solidity function isVCTemplateVerificationEnabled() public view returns (bool) |

* 入参：N/A
* 出参：开启返回true，关闭返回false
* 事件：N/A
* 核心逻辑：

1. 校验项目是否停用
2. 返回结果

3.1.3.10 **批量授权/撤销函数选择器**

批量授权/撤销不同链账户不同的函数选择器权限

批量撤销不同链账户各自的函数选择器权限，撤销哪个函数选择器传哪个

* 方法：

|  |
| --- |
| Solidity function batchOperateSelectorPermissions(  AccountSelector[] calldata accountSelectors,  bool isRevoke ) public |

* 入参：链账户函数选择器结构体列表，是否撤销
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity event SelectorPermissionOperated(address indexed account, bytes4 indexed selector, bool indexed isRevoked); |

* 核心逻辑：

1. 校验调用者是否为超级管理员，校验项目是否停用
2. 循环校验accounts地址为非0地址
3. 循环上链写入
4. 循环触发事件

3.1.3.11 **查询链账户的函数选择器权限**

查询一个链账户是否拥有某个函数选择器的访问权限

* 方法：

|  |
| --- |
| Solidity function hasSelectorPermission(address account, bytes4 selector) public view returns (bool) |

* 入参：链账户地址，函数选择器
* 出参：存在返回true，不存在返回false
* 事件：N/A
* 核心逻辑：

1. 校验项目是否停用
2. 校验accounts地址为非0地址
3. 返回结果

3.1.3.12 **查询链账户所有函数选择器**

查询一个链账户拥有的所有函数选择器

* 方法：

|  |
| --- |
| Solidity function getAllSelectorsForUser(address account) public view returns (bytes4[] memory selectorList) |

* 入参：链账户地址
* 出参：函数选择器列表
* 事件：N/A
* 核心逻辑：

1. 校验项目是否停用
2. 校验accounts地址为非0地址
3. 返回结果

3.1.3.13 **校验函数选择器写方法**

校验某一个写方法下某一个链账户是否拥有函数选择器权限

对外提供对应接口方法

* 方法：

|  |
| --- |
| Solidity function checkWriteFuncSelectorPermission(string memory method, address account, bytes4 selector) public view |

* 入参：did标识符里的方法名称，链账户地址，函数选择器
* 出参：N/A
* 事件：N/A
* 核心逻辑：

1. 校验项目是否停用
2. 校验当前入参method是否和设置的项目的\_method是否相同，不相同抛出异常并回滚交易
3. 校验accounts地址为非0地址
4. 如果是超级管理员直接返回
5. \_selectorPermissions中是否存在，不存在抛出异常并回滚交易

3.1.3.14 **校验函数选择器查方法**

校验某一个查方法下某一个链账户是否拥有函数选择器权限

对外提供对应接口方法

* 方法：

|  |
| --- |
| Solidity function checkQueryFuncSelectorPermission(address account, bytes4 selector) public view |

* 入参：链账户地址，函数选择器
* 出参：N/A
* 事件：N/A
* 核心逻辑：

1. 校验项目是否停用
2. 校验accounts地址为非0地址，是则抛出异常并回滚交易
3. 如果是超级管理员直接返回
4. 如果项目是公开项目直接返回
5. 如果项目是私有项目，则需要校验是否拥有该selector权限
6. \_selectorPermissions中是否存在，不存在抛出异常并回滚交易

3.1.3.15 **校验issuer审核状态权限**

校验某一个链账户是否在开启issuer审核状态权限下具有访问权限

对外提供对应接口方法

* 方法：

|  |
| --- |
| Solidity function checkIssuerVerificationEnabled(address account) public view |

* 入参：链账户地址
* 出参：N/A
* 事件：N/A
* 核心逻辑：

1. 如果开启发证方审核并且调用者不是超级管理员则抛出异常并回滚交易

3.1.3.16 **校验VC模版审核状态权限**

校验某一个链账户是否在开启VC模版审核状态权限下具有访问权限

对外提供对应接口方法

* 方法：

|  |
| --- |
| Solidity function checkVCTemplateVerificationEnabled(address account) public view |

* 入参：链账户地址
* 出参：N/A
* 事件：N/A
* 核心逻辑：

1. 如果开启VC模版审核并且调用者不是超级管理员则抛出异常并回滚交易

3.1.3.17 **项目启用**

openzeppelin官方包PausableUpgradeable合约启用方法，对应控制项目启用。因DID合约、Issuer合约、VC合约的读写方法都会经由checkQueryFuncSelectorPermission或checkWriteFuncSelectorPermission调用该合约的内部\_requireNotPaused()方法校验，所以控制这三个合约的启用也是这个方法

* 方法：

|  |
| --- |
| Solidity function pause() public |

* 入参：N/A
* 出参：N/A
* 调用方：门户
* 事件：

|  |
| --- |
| Solidity event Paused(address account) |

* 核心逻辑：

1. 校验调用者是否为超级管理员
2. 链上写入
3. 触发事件

3.1.3.18 **项目停用**

openzeppelin官方包PausableUpgradeable合约停用方法，对应控制项目停用。因DID合约、Issuer合约、VC合约的读写方法都会经由checkQueryFuncSelectorPermission或checkWriteFuncSelectorPermission调用该合约的内部\_requireNotPaused()方法校验，所以控制这三个合约的停用也是这个方法

* 方法：

|  |
| --- |
| Solidity function unpause() public |

* 入参：N/A
* 出参：N/A
* 调用方：门户
* 事件：

|  |
| --- |
| Solidity event Unpaused(address account) |

* 核心逻辑：

1. 校验调用者是否为超级管理员
2. 链上写入
3. 触发事件

3.1.3.19 **查询项目是否停用**

openzeppelin官方包PausableUpgradeable合约方法，查询合约是否停用，对应项目是否停用

* 方法：

|  |
| --- |
| Solidity function paused() public view virtual returns (bool) |

* 入参：N/A
* 出参：停用返回true，启用返回false
* 事件：N/A
* 核心逻辑：

1. 返回结果

3.2 **DID合约**

3.2.1 **功能介绍**

DID合约用于管理链上did信息。

3.2.2 **数据结构**

* did存储结构体

|  |
| --- |
| Solidity struct DidInfo {  string didDocument;  address account; // 记录链账户信息用于更新 } |

* did对应did信息映射

|  |
| --- |
| Solidity mapping(string => DidInfo) private \_didData; //key为did标识符 |

3.2.3 **方法定义**

3.2.3.1 **注册did**

注册did

* 方法：

|  |
| --- |
| Solidity 代码块 function registerDid(  string memory did,  string memory didDocument ) public |

* 入参：did标识符，did文档
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity 代码块 event DidRegisterd(string indexed did, string indexed didDocument); |

* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkWriteFuncSelectorPermission方法
2. 校验入参是否非空字符串，为空抛出异常并回滚交易
3. 校验did标识符是否存在，存在抛出异常并回滚交易
4. 上链写入
5. 触发事件

3.2.3.2 **更新did**

更新did文档

* 方法：

|  |
| --- |
| Solidity function updateDidDocument(  string memory did,  string memory didDocument ) public |

* 入参：did标识符，did文档
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity event DidDocumentUpdated(string indexed did, string indexed didDocument); |

* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkWriteFuncSelectorPermission方法
2. 校验入参是否非空字符串，为空抛出异常并回滚交易
3. 校验did标识符是否存在，不存在抛出异常并回滚交易
4. 获取原来DidInfo里account比对此次调用者的\_msgSender()是否相同，不同抛出异常并回滚交易
5. 上链写入
6. 触发事件

3.2.3.3 **查询did**

通过did标识符查出did文档

* 方法：

|  |
| --- |
| Solidity function getDidInfo(  string memory did ) public view returns (string memory didDocument) |

* 入参：did标识符
* 出参：did文档
* 事件：N/A
* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkQueryFuncSelectorPermission方法
2. 校验入参是否非空字符串，为空抛出异常并回滚交易
3. 返回结果

3.2.3.4 **校验did是否存在**

通过did标识符校验did是否上链存在

对外提供对应接口方法

* 方法：

|  |
| --- |
| Solidity function checkDid(  string memory did ) public view |

* 入参：did标识符
* 出参：N/A
* 事件：N/A
* 核心逻辑：

1. 校验入参是否非空字符串，为空抛出异常并回滚交易
2. 校验did对应信息是否存在，不存在抛出异常并回滚交易

3.3 **Issuer合约**

3.3.1 **功能介绍**

管理已被审核通过的发证方的基本信息和Vc模板。

3.3.2 **数据结构**

* Issuer存储结构体

|  |
| --- |
| Solidity 代码块 struct IssuerInfo {  string name;//发证方名称  bool isDisabled;//是否禁用  address account; // 记录链账户信息用于更新 } |

* 发证方did标识符对应发证方信息映射

|  |
| --- |
| Solidity 代码块 mapping(string => IssuerInfo) private \_issuerInfoData; //key为发证方Did |

* 模板存储结构体

|  |
| --- |
| Solidity 代码块 struct VcTemplateInfo {  string vcTemplateData;  address account; // 记录链账户信息用于更新  bool isDisabled;//是否禁用 } |

* 模版编号对应模版信息映射

|  |
| --- |
| Solidity mapping(string => VcTemplateInfo) private \_vcTemplateInfoData; //key为模板编号； |

* 发证方名称对应bool映射

|  |
| --- |
| Solidity mapping(string => bool) private \_issuerName; //用于校验发证方名称是否唯一 |

3.3.3 **方法定义**

3.3.3.1 **注册发证方**

注册发证方

* 方法：

|  |
| --- |
| Solidity 代码块 function registerIssuer(  string memory issuerDid,  string memory name ) public |

* 入参：发证方did，发证方名称，项目编码
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity 代码块 event IssuerRegisterd(  string indexed issuerDid,  string indexed name,  bool indexed isDisabled ) |

* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkIssuerVerificationEnabled方法
2. 调用IFunctionSelectorACL合约的checkWriteFuncSelectorPermission方法
3. 校验入参是否非空字符串，为空抛出异常并回滚交易
4. 调用IDid合约的checkDid方法
5. 校验name是否唯一，不唯一抛出异常并回滚交易
6. 校验发证方信息是否存在，存在抛出异常并回滚交易
7. 上链写入
8. 触发事件

3.3.3.2 **更新发证方**

更新发证方名称，是否禁用

* 方法：

|  |
| --- |
| Solidity 代码块 function updateIssuer(  string memory issuerDid,  string memory name ) public |

* 入参：发证方did，发证方名称
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity 代码块 event IssuerUpdated(string indexed issuerDid, string indexed name, bool indexed isDisabled) |

* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkIssuerVerificationEnabled方法
2. 调用IFunctionSelectorACL合约的checkWriteFuncSelectorPermission方法
3. 校验入参是否非空字符串，为空抛出异常并回滚交易
4. 校验发证方信息是否存在，不存在抛出异常并回滚交易
5. 获取原来数据里的name比对入参name是否相同，相同抛出异常并回滚交易，不同校验name是否唯一，不唯一

抛出异常并回滚交易

1. 获取原来数据里的account比对此次调用者的\_msgSender()是否相同，不同抛出异常并回滚交易
2. 上链写入
3. 触发事件

3.3.3.3 **启停发证方**

启用或停用发证方

* 方法：

|  |
| --- |
| Solidity 代码块 function changeIssuerStatus(  string memory issuerDid,  bool isDisabled ) public |

* 入参：发证方did，是否停用
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity 代码块 event IssuerStatusChanged(string indexed issuerDid, bool indexed isDisabled) |

* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkIssuerVerificationEnabled方法
2. 调用IFunctionSelectorACL合约的checkWriteFuncSelectorPermission方法
3. 校验入参是否非空字符串，为空抛出异常并回滚交易
4. 校验发证方信息是否存在，不存在抛出异常并回滚交易
5. 获取原来数据里的account比对此次\_msgSender()是否相同，不同抛出异常并回滚交易
6. 上链写入
7. 触发事件

3.3.3.4 **查询发证方信息**

通过发证方did查询发证方信息

* 方法：

|  |
| --- |
| Solidity 代码块 function getIssuerInfo(  string memory issuerDid ) public view returns (string memory, bool) |

* 入参：发证方did
* 出参：发证方名称，是否停用
* 事件：N/A
* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkIssuerVerificationEnabled方法
2. 调用IFunctionSelectorACL合约的checkQueryFuncSelectorPermission方法
3. 校验入参是否非空字符串，为空抛出异常并回滚交易
4. 返回结果

3.3.3.5 **校验发证方是否存在**

校验发证方did标识符是否存在

对外提供对应接口方法

* 方法：

|  |
| --- |
| Solidity function checkIssuer(  string memory issuerDid ) public view |

* 入参：发证方did标识符
* 出参：N/A
* 事件：N/A
* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkQueryFuncSelectorPermission方法
2. 校验入参是否非空字符串，为空抛出异常并回滚交易
3. 校验did对应发证方信息是否存在，不存在抛出异常并回滚交易

3.3.3.6 **注册VC模板**

注册VC模板

* 方法：

|  |
| --- |
| Solidity 代码块 function registerVCTemplate(  string memory vcTemplateId,  string memory vcTemplateData ) public |

* 入参：VC模板id，VC模板数据
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity 代码块 event VCTemplateRegisterd(  string indexed vcTemplateId,  string indexed vcTemplateData ) |

* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkVCTemplateVerificationEnabled方法
2. 调用IFunctionSelectorACL合约的checkWriteFuncSelectorPermission方法
3. 校验入参是否非空字符串，为空抛出异常并回滚交易
4. 校验Vc模版信息是否存在，存在抛出异常并回滚交易
5. 上链写入
6. 触发事件

3.3.3.7 **更新VC模板**

更新VC模板

* 方法：

|  |
| --- |
| Solidity 代码块 function updateVCTemplate(  string indexed vcTemplateId,  string indexed vcTemplateData ) public |

* 入参：VC模板id，VC模板数据
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity 代码块 event VCTemplateUpdated(  string indexed vcTemplateId,  string indexed vcTemplateData ) |

* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkVCTemplateVerificationEnabled方法
2. 调用IFunctionSelectorACL合约的checkWriteFuncSelectorPermission方法
3. 校验入参是否非空字符串，为空抛出异常并回滚交易
4. 校验vc模版信息是否存在，不存在抛出异常并回滚交易直接返回
5. 获取原来数据里account比对此次调用者的\_msgSender()是否相同，不同抛出异常并回滚交易
6. 上链写入
7. 触发事件

3.3.3.8 **启停VC模板**

启停VC模板

* 方法：

|  |
| --- |
| Solidity 代码块 function changeVCTemplateStatus(  string memory vcTemplateId,  bool isDisabled, ) public |

* 入参：VC模板id，是否停用
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity 代码块 event VCTemplateStatusChanged(  string indexed vcTemplateId,  bool indexed isDisabled ) |

* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkVCTemplateVerificationEnabled方法
2. 调用IFunctionSelectorACL合约的checkWriteFuncSelectorPermission方法
3. 校验入参是否非空字符串，为空抛出异常并回滚交易
4. 校验vc模版信息是否存在，不存在抛出异常并回滚交易直接返回
5. 获取原来数据里account比对此次\_msgSender()是否相同，不同抛出异常并回滚交易
6. 上链写入
7. 触发事件

3.3.3.9 **查询VC模板信息**

通过VC模板Id查询VC模版数据，是否停用

* 方法：

|  |
| --- |
| Solidity 代码块 function getVCTemplateInfo(  string memory vcTemplateId ) public view returns (string memory, bool); |

* 入参：VC模板Id
* 出参：VC模版数据，是否停用
* 事件：N/A
* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkVCTemplateVerificationEnabled方法
2. 调用IFunctionSelectorACL合约的checkQueryFuncSelectorPermission方法
3. 校验入参是否非空字符串，为空抛出异常并回滚交易
4. 返回结果

3.4 **VC合约**

3.4.1 **功能介绍**

VC合约用于记录存证，以及吊销存证。

3.4.2 **数据结构**

* VC存证信息结构体

|  |
| --- |
| Solidity 代码块 struct VCInfo {  string vcId;  string issuerDid; //发证方Did  string vcHash;  bool isRevoked //是否已吊销 } |

* VcId对应vc存证信息映射

|  |
| --- |
| Solidity mapping(string => VCInfo) private \_vcInfoData; //key为vcId |

3.4.3 **方法定义**

3.4.3.1 **创建VC**

创建存证

* 方法：

|  |
| --- |
| Solidity 代码块 function createVC(  string memory vcId，  bytes32 vcHash,  string memory issuerDid ) public |

* 入参：存证 id，VCHash，发证方did
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity 代码块 event VCCreated(  string indexed vcId,  bytes32 indexed vcHash,  string indexed issuerDid ); |

* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkWriteFuncSelectorPermission方法
2. 校验入参是否非空字符串，为空抛出异常并回滚交易
3. 调用IIssuer合约的checkIssuer方法
4. 校验发证方信息是否存在，不存在抛出异常并回滚交易直接返回
5. 校验VC存证是否存在，存在抛出异常并回滚交易直接返回
6. 上链写入
7. 触发事件

3.4.3.2 **查询VC**

通过VCId查询VCHash

* 方法：

|  |
| --- |
| Solidity function getVCHash(  string memory vcId ) public view returns (string memory) |

* 入参：VC存证id
* 出参：VCHash
* 事件：N/A
* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkQueryFuncSelectorPermission方法
2. 校验入参是否非空字符串，为空抛出异常并回滚交易
3. 返回结果

3.4.3.3 **吊销VC**

对一个vc存证进行吊销

* 方法：

|  |
| --- |
| Solidity 代码块 function revokedVC(  string memory vcId,  bool isRevoked ) public |

* 入参：VC存证id，是否吊销
* 出参：N/A
* 事件：

|  |
| --- |
| Solidity 代码块 event VCRevoked(  string indexed vcId,  bool indexed isRevoked ) |

* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkWriteFuncSelectorPermission方法
2. 校验入参是否非空字符串，为空抛出异常并回滚交易
3. 校验VC存证是否存在，不存在抛出异常并回滚交易直接返回
4. 上链写入
5. 触发事件

3.4.3.4 **查询吊销状态**

通过vc存证id查询吊销状态

* 方法：

|  |
| --- |
| Solidity 代码块 function getVCRevokedStatus(  string memory vcId ) public view returns (bool) |

* 入参：VC存证id
* 出参：吊销状态
* 事件：N/A
* 核心逻辑：

1. 调用IFunctionSelectorACL合约的checkQueryFuncSelectorPermission方法
2. 校验入参是否非空字符串，为空抛出异常并回滚交易
3. 返回结果