1. 引言

1.1. 编写目的

该合约旨在提供一个普适性的合规审计方案，能够事先于智能合约中自定义在审计过程中的一系列检查项，使得事务执行者能够在规则约束下”合法“参与。

1.2. 背景

目前的区块链落地主要基于区块链不可篡改的属性应用于存证、溯源等场景，而不可窜改性事实上是第一代区块链（如比特币）已经具备的，如何能够基于智能合约为各行业赋能是区块链应用面临的一个重大课题。

在做业务探索时我们按如下思路思考：运行于区块链虚拟机上的智能合约是区块链账本所承载的信任的自然传递，而连接链上与链下的桥梁——预言机服务可以看做是区块链的”眼睛“。如果能事先定义好规则让预言机服务实时”看到“现实世界中的事务，根据人们的行为智能决策并将这些行为以不可篡改的事件形式记录在区块链上，就能形成”事前定义“、”事中监控“、”事后问责“的完整业务闭环。

1.3. 需求与痛点

在银行中存在这样一个真实业务场景，银行的某些工作人员需要在固定的时间周期（如：每个月两次）去核算、检查与各商户的POS机业务情况。但银行无法确定其工作人员是否去现场真实执行该业务，尽管随着通信技术的发展可以要求工作人员拍摄到场视频等作为佐证，但业务中可操作性很大难以保证工作人员造假，而且还存在工作人员与业务流程管理员串通修改风险。

如果将合规审计方案应用到改POS审计场景，我们可以准备用于验证时间、地点、人脸识别以及物体识别等预言机服务，并事先定义规则以限定时间（如：工作日的上班时间）、地点（如指定商户坐标的1千米范围内）、须本人操作（通过人脸识别服务与预先录入的该工作人员的人脸特征信息进行比对验证）且识别到指定类型的POS机（通过物体识别服务与预先录入的POS机型特征信息进行比对验证）等约束条件，并注册到审计智能合约中。这样当银行工作人员执行该业务时，会在审计智能合约中登记审计事件，只有符合上述左右约束条件才能验证成功。可以看到我们应用区块链技术不仅有效监督银行业务人员现场的业务执行情况，而且利用区块链不可篡改的属性防止银行内部人员传统作假，并建立起可靠的事后问责机制。

1.4. 核心价值

运用视频图像类算法与区块链技术，对金融合规类业务进行智能识别，在时间、空间等范围条件下监控跟踪事件的执行过程，通过将审计合规工作进行上链，保障合规审计工作的规范性、安全性、时空性、便捷性。主要价值体现如下：

- 基于时空的智能合约：按照事先制定的业务规则制定合约，保证合约在一定的时间和空间内执行，并记录上链。
- 安全的人脸验证：利用生物识别算法并基于密码学的用户信息摘要验证、个人信息加密存储、以及零知识证明相关技术，保障在不泄露用户隐私的情况下进行身份验证。
- 提高合规审计工作效率与真实性：分布式账本记账，数据不可篡改可追溯，安全可信任，实时审计，节省繁琐的审计工作成本

在本次提交的智能合约实现中，我们设计了可应用于所有合规审计业务的普适性方案，并针对上述银行业务场景中所需的预言机服务以及规则方案做了细化。其他的审计场景可以视具体业务做类似的细化实现。

1.5. 定义

- **合约运维人员（maintainer）**：保障合约正常运转的运维人员，具有录入审计当事人、规则、规则方案、项目等信息的权限

- **审计当事人（auditee）**：审计业务中的执行人员，如上文提到的银行工作人员
- **规则（rule）**：用于定义一项具体的约束验证，规则会有类别，若验证需要涉及与区块链无关的上下文，一般情况下该类规则会对应一个预言机服务
- **规则方案（rule schema）**：由一个或多个规则组成，用于表达审计当事人在某个具体业务场景下需要验证的所有规则
- **项目（project）**：定义一个具体的业务，并包含该业务中所涉及的所有审计当事人以及规则方案。当然我们将不同的业务部署在不同的智能合约中，这里抽象出项目的概念是为了复用审计当事人以及规则方案等，为业务实现人员提供便利
- **合规事件（event）**：定义审计当事人的一次具体的业务执行记录

2. 数据存储环境说明

合规审计业务涉及的数据存储可分为链上和链下存储两个部分：

2.1. 链上存储

由于该审计方案在多个联盟链上编写，为了统一存储方式，所有的链上数据存储均采用键值存储的方式。

Fabric和XuperChain在智能合约上下文中都提供了键值存储的接口，在这两个联盟链的智能合约实现中，我们一般使用Key值存储一个对象的ID，而Value值用于存储对象本身。考虑到对象信息的可读性，我们采用了JSON序列化的方式存储。

目前在CITA和FISCO BCOS平台都使用solidity语言开发，两个平台上数据存储统一使用solidity原生的`mapping`存储，考虑到平台之间的统一性我们未在FISCO BCOS平台使用Table存储数据。与Fabric和XuperChain类似，我们使用Key值存储一个对象的ID，而Value值用于存储对象本身。

2.2. 链下存储

当我们注册规则时需要给定规则约束条件，这些约束条件会在注册规则项时传入或者在构建预言机服务的时候内置于其中，而上述约束条件以及对应的规则ID映射关系都存储于预言机服务附带的存储中。

3. 数据结构的命名规则

这里先给出代码中用到的命名规则说明：

- **蛇形命名**：所有单词小写，单词与单词之间使用下划线分割，如`snak_style`
- **首字母大写驼峰命名**：所有单次的首字母都大写，如`CamleStyle`
- **首字母小写驼峰命名**：首字母小写，随后的单词首字母大写，如`camleStyle`
- **所有字母大写**：如`CAMLESTYLE`

目前Fabric和XuperChain平台上的智能合约通过go语言实现，而CITA和FISCO BCOS平台使用solidity语言开发，为了保持所有平台之间语言风格的一致性且不破坏不同语言固有的命名规则，定义命名规则如下：

3.1. go语言命名规则

3.1.1 目录管理

go语言目录组织符合GOPATH组织规范，所有源码文件位于src目录下。目录名使用蛇形命名法。

src下的core子目录用于存储Fabric和XuperChain平台的共有逻辑，而平台相关的逻辑分别存储在fabric和xchain子目录内。

3.1.2. 包管理

考虑到fabric1.4版本对go mod并未很好地支持，我们在go语言开发中采用GOPATH方式进行包管理。为了方便代码阅读我们在导入智能合约的内部包时使用了相对路径，而引入Fabric和XuperChain平台相关依赖时使用了绝对路径的方式。

这就要求我们在构建合约时不仅引入全局GOPATH，而且要将源码所在的目录作为GOPATH路径导入。不过我们为Fabric和XuperChain平台分别编写了Makefile脚本，使用make命令时无须手动设置GOPATH路径。

3.1.3. 文件命名

文件名一律使用蛇形命名法，与go语言对单元测试文件的命名要求保持一致。

3.1.4. 结构体命名

- 包内定义的结构体使用首字母小写驼峰命名法
- 需要在包外使用的结构体使用首字母大写驼峰命名法

3.1.5. 函数命名

- 仅在包内使用的函数使用首字母小写驼峰命名法
- 需要在包外使用的函数使用首字母大写驼峰命名法

3.1.6. 变量命名

- 仅在包内使用的全局变量使用首字母小写驼峰命名法
- 需要在包外使用的全局变量使用首字母大写驼峰命名法
- 通过const变量模拟枚举（Enum）的变量采用所有字母大写的命名方式
- 函数参数变量、本地变量使用首字母小写驼峰命名法，为了符合go语言的命名习惯，应尽量使用短小的变量名

3.2. solidity语言命名规则

3.2.1 目录管理

考虑到FISCO BCOS合约部署的便捷性，我们在使用solidity语言开发时采用了尽量扁平化的目录组织方式，目前除了用于存放通用接口的子目录interface之外，其他文件都直接存放在其所属的智能合约目录下。

3.2.2. 文件命名

文件名使用首字母大写驼峰命名法，这样与合约名保持一致为合约部署带来便利。

3.2.3. 合约命名

合约名使用首字母大写驼峰命名法。

3.2.4. 结构体命名

结构体名使用首字母大写驼峰命名法。

3.2.5. 方法命名

方法名使用首字母小写驼峰命名法，与构造方法命名规则保持一致。

3.2.6. 变量命名

- 结构体中的成员变量使用首字母大写驼峰命名法
- 其他变量一律使用首字母小写驼峰命名法

4. 安全性设计

4.1. 权限控制

在审计合约中主要涉及两类角色：合约运维人员和审计当事人。

合约运维人员独有的权限有：

- 录入规则
- 录入规则方案
- 录入审计当事人
- 录入项目

审计当事人独有的权限有：

- 新增审计事件

角色间共有的权限有：

- 列出所有的合约运维人员
- 查询规则方案
- 查询审计当事人
- 查询项目
- 查询某个当事人在某个项目中已登录的所有事件

由于所有联盟链框架都存在CA认证模块（这也是联盟链或者说特许链区别于公链的最大特点之一），因此权限控制相关的业务会通过CA认证模块实现，审计智能合约中的代码逻辑不做体现。

4.2. 有效性验证

当执行合约接口调用时，我们对参数进行了基本的有效性验证验证，当然这些有效性验证不足以保证调用安全，还需要上游负责数据收集的SDK，以及下游具体解析规则信息的预言机服务的配合。

5. 优化

优化对象（目标）	措施	效果
规则、规则方案的表达方式	自定义DSL语言	目前通过多个参数实现规则的注册及验证，通过自定义DSL实现更丰富的规则及规则值的表达方式。此外目前规则方案只支持逻辑操作符和具体规则项定义两层组合关系，可通过DSL实现更灵活的嵌套组合。
	定义	

预言机接口	预言机接口合约	由于业务需求多种多样，相应的验证方式也各不相同，为了有效管理用于验证的各预言机服务需要在后期建立预言机接口合约。该合约负责预言机服务的接入，通过定义统一的接口、发放证书等方式规范预言机服务。
权限认证	智能合约中引入权限认证	目前审计合约完全依赖联盟链中的权限控制模块，但由于智能合约与预言机服务直接交互，为了保障预言机服务本身后续也需要引入权限控制，因此需要在智能合约中引入权限认证用于保障自身以及与预言机的安全运转。
查询结果展示	自定义显示方案	目前查询结果处理中简单的将整个结果转换成json显示，有些字段在查询结果不能清晰的展示，如审计事件ID显示等。后续可以考虑针对不同的字段做单独的类型转换后显示。
数据存储	优化序列化方案	考虑到多个链的合约开发任务量，目前合约数据存是以开发速度优先来处理的，并没有考虑数据大小对链性能的影响，后续可以优化数据存储大小。

6. 数据结构设计

注：由于合约开发时从go语言开始的，下文中的定义是依照go语言给出的。由于go语言与solidity语言之间存在很多语法差异，solidity中的表达会和go语言存在一些差异，如果存在分歧以go语言为准。

6.1. 合约运维人员

由于不同的区块链中的人员组织结构设计情况不同，为了更好地兼容不同智能合约，可以将合约运维人员定义放在合约中。合约运维人员可以是一个人也可以一组人，为了有更好的灵活性，可以通过数组的方式维护。

以下为单个运维人员的结构定义：

序号	名称	类型	备注
1	ID	uint32	唯一ID，由智能合约分配
2	Name	string	名称

6.2. 审计当事人

由于合约调用一般发生在受限人群，因此需要实现由合约运维人员录入当事人的信息，后续当事人提交事务时会对当事人的身份有效性进行验证。

以下为当事人的结构定义：

序号	名称	类型	备注
1	ID	uint32	唯一ID，由智能合约分配
2	Name	string	名称

6.3. 规则

我们可能需要处理各类的审计需求，因此需要对规则也进行定义。

限于时间等因素，目前的规则仅限于时间、定位（空间）、人脸识别、物体识别几个特定项，后续可以考虑通过DSL（Domain Secific Language）来定义。

6.3.1. 规则类型

序号	名称	值	备注
1	无	0x00	无规则占位符，没有规则意味无任何限制
2	时间	0x01	对应时间预言机的规则类型
3	定位	0x02	对应定位预言机的规则类型
4	人脸识别	0x03	对应人脸识别预言机的规则类型
5	物体识别	0x04	对应物体识别预言机的规则类型

6.3.2. 注册规则

序号	名称	类型	备注
1	ID	uint32	唯一ID，由预言机服务分配
2	Type	规则类型	6.3.1定义的规则类型枚举值
3	Expression	string	规则的验证表达式

6.3.3. 验证规则

序号	名称	类型	备注
1	ID	uint32	唯一ID，对应6.3.2中的规则ID
2	Type	规则类型	6.3.1定义的规则类型枚举值
3	ActualValue	string	符合规则验证表达式要求的验证值

6.3.4 条件表达式

序号	名称	值	备注
1	GT	>	
2	GE	>=	
3	LT	<	
4	LE	<=	
5	EQ	==	
6	IN	in	in标识在一个范围内，可以是两个数值之间的范围，如1到10之间；也可以是一个集合范围，比如可以是1、2、3、4中的任何一个整数

6.4. 规则方案

规则方案由逻辑关系操作符，以及一个或多个规则组成

6.4.1. 逻辑关系操作符

序号	名称	值	备注
1	NONE	NONE	表示空规则
2	NOT	NOT	规则验证结果取非
3	AND	AND	必须所有规则都满足
4	OR	OR	只要其中一个规则满足即可

6.4.2. 规则方案结构

序号	名称	类型	备注
1	ID	uint32	唯一ID，由智能合约分配
2	Operator	逻辑关系操作符	6.4.1定义的逻辑关系操作符枚举值
3	Rules	map	用于记录一组规则，Key值对应一个规则类型，Value值为注册规则表达式时预言机服务返回的相应的ID值

6.5. 审计业务

尽管我们可以针对一个审计的业务单独创建一个智能合约，不过考虑到如果有很强关联性的业务，若是业务中涉及的人员或者规则有很大的重叠，能够在同一个智能合约中表达多个审计业务会更有利于开发部署。

以下给出一个审计业务的定义：

序号	名称	类型	备注
1	ID	uint32	业务唯一ID
2	Name	string	名称
3	Description	string	业务的简单描述信息
4	AuditeeRulesMap	map	用于记录执行该审计业务的一组审计当事人以及其对应的规则方案。 其中Key值对应一个审计当事人，Value值为规则方案。

6.6. 审计事件

6.6.1. 审计当事人规范

序号	名称	类型	备注
1	ID	uint256	规范ID，前12byte为审计当事人ID+项目ID+规则ID，多余字节可支持之后扩展
2	Auditee	审计当事人结构	可参见6.2对审计当事人的定义
3	Project	审计业务结构	可参见6.5对审计业务的定义
4	Rule	规则方案结构	可参见6.4.2对规则方案的定义

6.6.2. 合规事件登录

序号	名称	类型	备注
1	Index	uint32	用于标记隶属于一个业务下该审计当事人第几次录入
2	AuditeeSpecification	审计当事人规范结构	可参见6.6.1对审计当事人规范的定义
3	Timestamp	uint64	登录事件发生时的时间戳
4	Params	[]string	登录涉及的参数