私有数据

注解

本主题假设你已经理解了在 私有数据文档 中所描述概念。

私有数据集合定义

一个集合定义包含了一个或者多个集合，每个集合具有一个策略列出了在集合中的所有组织，还包括用来控制在背书阶段私有数据的传播所使用的属性，另外还有一个可选项来决定数据是否会被删除。

从 Fabric 2.0中引入的 Fabric 链码生命周期开始 集合定义是链码定义的一部分。集合是 由通道成员批准，然后在链码定义提交到通道上时被部署。集合文件需要对所有通道成员保持一致。 如果你使用 peer CLI 工具来批准和提交 链代码定义，使用 `--collections-config` 标签来指定集合定义文件的路径。 如果你正在使用 Node.js 的 Fabric SDK， 访问 如何安装和启动链码。 当 初始化链码时，使用 之前的生命周期流程 来部署私有数据集合， 使用 `--collections-config` 标签。

集合定义由下边的属性组成：

- `name`: 集合的名字.
- `policy`: 私有数据集合分发策略，它通过 `Signature` 策略语法定义了允许哪些组织的 Peer 节点持久化集合数据。为了支持读/写交易，私有数据的分发策略必须要定义一个比链码背书策略更大范围的一个组织的集合，因为 Peer 节点必须要拥有这些私有数据才能来对这些交易提案进行背书。比如，在一个具有10个组织的通道中，其中5个组织可能会被包含在一个私有数据集合的分发策略中，但是背书策略可能会调用其中任意的3个组织来进行背书。
- `requiredPeerCount`: 在节点为背书签名并将提案响应返回给客户端前，每个背书节点必须将私有数据分发到的节点(在被授权的组织当中)的最小数量。将传播作为背书的一个条件可以确保即使背书背书不可用，私有数据在网络中也还是可用的。 `requiredPeerCount` 设为 `0` 代表分发并不是 **必须** 的，但是如果 `maxPeerCount` 比0大的话，就需要分发。通常不建议 `requiredPeerCount` 设为 `0` 通常，因为那会造成在背书节点不可用的时候，网络中的私有数据可能会丢失。通常在背书的时候你会希望分发私有数据到多个节点以保证网络中私有数据的冗余存储。
- `maxPeerCount`: 为了数据的冗余存储，每个背书节点将会尝试将私有数据分发到的其他节点（在被授权的组织中) 的最大数量。如果在背书和提交之间一个背书节点不可用了，其他节点就可以在背书的时候从已经收到私有数据的节点拉取私有数据。如果这个值被设置为 `0`，私有数据在背书的时候就不会被分发，这会在提交的时候强制节点从授权的背书节点上拉取私有数据。
- `blockToLive`: 代表了数据应该以区块的形式在私有数据库中存在多久。数据会在私有数据库上存在指定数量个区块，然后它会被删除，然后链码就无法查询到该数据，其他节点也无法请求到该数据。如果要无限期的保持私有数据，也就是从来不删除私有数据的话，将 `blockToLive` 设置为 `0`。
- `memberOnlyRead`: `true` 值表示节点自动会强制只有属于这些集合的组织的客户端才可以读取私有数据。如果一个非成员组织的客户端试图执行一个链码方法来读取私有数据的话，会结束链码的调用并产生错误。如果你想在单独的链码方法中进行更细粒度的访问控制的话，可以使用 `false` 值。
- `memberOnlyWrite`: `true` 值表示 peer 节点自动 强制仅属于集合成员组织之一的客户端 允许从链码写入私有数据。如果来自非成员组织的客户 试图执行在私有数据键上执行写操作的链码函数， 链码

调用因错误而终止。利用价值 如果您想在中编码更细粒度的访问控制，请选择“false” 单个链码函数，例如，您可能需要某些客户端 从非成员组织到能够在某个 收藏。`true` 值表示 peer 节点自动强制只允许属于集合成员组织之一的客户端从链码写入私有数据。 如果来自非成员组织的客户端 试图执行在私有数据键上执行写操作的链码函数， 链码调用因错误而终止。 利用 `false` 值 如果你想在单独的链码方法中编码更细粒度的访问控制， 例如你可能希望来自非成员组织的特定的客户端能在特定的集合中创建私有数据， 请使用``false``。

- `endorsementPolicy`: 一种可供选择的背书策略 用来覆盖链码级别背书策略的集合。一个 集合级别的背书策略可能以 `signaturePolicy` 或可能是 `channelConfigPolicy` 引用到一个通道配置的已存在策略的形式指定。 `endorsementPolicy` 可能与集合分发 `policy` 相同，也可能需要 较少或额外的组织 peer。

下边是一个集合定义的 JSON 文件示例，一个包含了两个集合定义的数组：

```
[
 {
    "name": "collectionMarbles",
    "policy": "OR('Org1MSP.member', 'Org2MSP.member')",
    "requiredPeerCount": 0,
    "maxPeerCount": 3,
    "blockToLive":1000000,
    "memberOnlyRead": true,
    "memberOnlyWrite": true
 },
 {
    "name": "collectionMarblePrivateDetails",
    "policy": "OR('Org1MSP.member')",
    "requiredPeerCount": 0,
    "maxPeerCount": 3,
    "blockToLive":3,
    "memberOnlyRead": true,
    "memberOnlyWrite":true,
    "endorsementPolicy": {
      "signaturePolicy": "OR('Org1MSP.member')"
    }
 }
]
```

此示例使用来自 Fabric 测试网络的组织，`Org1` 和 `Org2`。`collectionMarbles` 定义中的策略授权两个组织都能访问私有数据。当 链码数据需要从排序服务节点保持私有时，这是一个典型的配置。 但是，`collectionMarblePrivateDetails` 定义中的策略限制访问 到通道中的组织子集（在这种情况下为 `Org1`）。 此外， 写到这个集合需要来自 `Org1` peer 节点的背书，甚至 虽然链码级别的背书策略可能需要``Org1`` 或` *Org2``的背书。 由于“memberOnlyWrite”为 true，只有 ``Org1*`` 的客户端 可以调用写入私有数据集合的链码。 这样你就可以控制哪些组织被委托写入特定的私有数据集合。 私人数据集合。

私有数据分发

由于私有数据不会被包含在提交到排序服务的交易中，因此也就不会被包含在区块中，背书节点扮演着将私有数据分发给其他授权组织的节点的重要角色。这确保了私有数据在背书节点完成背书之后变成不可用的时候的可用性。为了辅助分发，在集合定义中的 `maxPeerCount` 和 `requiredPeerCount` 属性控制了在背书的时候分发的数量。

如果背书节点不能够成功地将私有数据分发到至少 `requiredPeerCount` 的要求，它将会返回一个错误给客户端。背书节点会尝试将私有数据分发到不同组织的节点，来确保每个被授权的组织具有私有数据的

一个副本。因为交易在链码执行期间还没有被提交，背书节点和接收节点除了在它们的区块链之外，还在一个本地的 `临时存储（transient store）` 中存储了一个私有数据副本，直到交易被提交。

当一个被授权的节点在提交的时候，如果他们的临时存储中没有私有数据的副本（或者是因为他们不是一个背书节点，或者是因为他们在背书的时候没有接收到私有数据），他们会尝试从其他的被授权的节点那里拉取私有数据，尝试会*持续一个可配置的时间长度*，在时间可以通过节点配置文件 `core.yaml` 中的属性 `peer.gossip.pvtData.pullRetryThreshold` 进行配置。

注解

只有当提出请求的节点是私有数据分发策略定义的集合中的一员的时候，被询问的节点才会返回私有数据。

当使用 `pullRetryThreshold` 时候需要考虑的问题：

- 如果提出请求的节点能够在 `pullRetryThreshold` 时间内拿到私有数据的话，它将会把交易提交到自己的账本（包括私有数据的哈希值），并且将私有数据存储在与其他的通道状态数据进行了逻辑隔离的状态数据库中。
- 如果提出请求的节点没能在 `pullRetryThreshold` 时间内拿到私有数据的话，它将会把交易提交到自己的账本（包括私有数据的哈希值），但是不会存储私有数据。
- 如果某个节点有资格拥有私有数据，却没有得到的话，这个节点就无法为将来会引用这个丢失的私有数据的交易进行背书，背书时会发现无法查询到键 （基于在状态数据库中主键的哈希值），并且链码将会收到一个错误。

因此，将 `requiredPeerCount` 和 `maxPeerCount` 设置成足够大的值来确保在你的通道中的私有数据的可用性是非常重要的。比如，如果在交易提交之前，每个背书节点都不可用了，`requiredPeerCount` 和 `maxPeerCount` 属性将会确保私有数据在其他的节点上是可用的。

注解

为了让集合能够工作，正确配置跨组织的 gossip 非常重要的。请阅读 Gossip 数据传播协议，尤其注意“锚节点”和“外部端点”配置。

从链码中引用集合

我们可以用 shim API 设置和取回私有数据。

相同的链码数据操作也可以应用到通道状态数据和私有数据上，但是对于私有数据，要在链码 API 中指定和数据相关的集合的名字，比如 `PutPrivateData(collection,key,value)` 和 `GetPrivateData(collection,key)`。

一个链码可以引用多个集合。

引用链码中的隐式集合

从 v2.0 开始，通道中的每一个组织都可以使用隐式私有数据集合， 这样如果你想使用每个组织的集合，就不必定义集合了。 每个特定 org 的隐式集合 具有匹配组织的分配策略和背书策略。 因此，你可以将隐式集合用于你想要的用例 以确保特定组织已写入集合键命名空间。 v2.0 链码生命周期使用隐式集合来跟踪哪些组织 已经批准了链码定义。 类似地，你可以在应用链码中使用隐式集合 来跟踪哪些组织已批准或投票 状态的变化。

若要写入和读取隐式私有数据集合键，请在 `PutPrivateData` 并 `GetPrivateData` 链码 API 中，指定集合参数为 `"_implicit_org_<MSPID>"`，例如 `"_implicit_org_Org1MSP"`。

如何在链码建议中传递私有数据

因为链码提案被存储在区块链上，不要把私有数据包含在链码提案中也是非常重要的。在链码提案中有一个特殊的字段 `transient`，可以用它把私有数据来从客户端（或者链码将用来生成私有数据的数据）传递给节点上的链码调用。链码可以通过调用 GetTransient() API 来获取 `transient` 字段。这个 `transient` 字段会从通道交易中被排除。

保护私有数据内容

如果私有数据相对简单并且可预测（例如，交易金额的数量）， 没有被授权给私有数据集合的通道成员可以通过暴力计算域名空间的 hash 来猜测私有数据的内容， 希望找到在链上找到和私有数据 hash 值匹配的数据。因此可预测的私有数据应该包含一个随机的 和私有数据键连接并且包含在私有数据值中的”salt“，所以匹配的 hash 不能真实地通过暴力计算找到。 随机 “salt” 可以在客户端生成（例如在安全的伪随机源取样）并且然后在链码调用时和私有数据在临时字段中一起传递。

私有数据的访问控制

在版本1.3之前，基于集合成员资格的对私有数据的访问控制 仅对 peer 强制实施。基于链码提案提交者组织的访问控制 要求用链码逻辑编码。集合配置选项 ``memberOnlyRead``（从 v1.4 版开始） 和 ``memberOnlyWrite``（自 v2.0 版以来）可以自动强制链码 提案提交者必须来自集合成员，才能进行读取或写入 私有数据键。有关集合配置定义的更多信息 以及如何设置它们，请参考 本文的`私有数据集合定义`_部分。

..注意:: 如果你想要更精细的访问控制，你可以设置 `memberOnlyRead` 和 `memberOnlyWrite` 设置为 false。然后你可以应用你 在链码中的访问控制逻辑，例如通过调用 GetCreator() 链码 API 或使用客户端身份 chaincode library 。

查询私有数据

私有集合数据能够像常见的通道数据那样使用 shim API 来进行查询：

- `GetPrivateDataByRange(collection, startKey, endKey string)`
- `GetPrivateDataByPartialCompositeKey(collection, objectType string, keys []string)`

对于 CouchDB 状态数据库，可以使用 shim API 查询 JSON 内容：

对于 CouchDB 状态数据库，JSON 内容查询可以使用 shim API 来传递：

- `GetPrivateDataQueryResult(collection, query string)`

限制：

- 客户端调用执行范围查询或者富查询链码的时候应该知道，根据上边关于私有数据分发部分的解释，如果他们查询的节点有丢失的私有数据的话，他们可能会接收到结果集的一个子集。客户端可以查询多个节点并且比较返回的结果，以确定一个节点是否丢失了结果集中的部分数据。
- 不支持在单个交易中既执行范围查询或者富查询并且更新数据，因为查询结果无法在以下类型的节点上进行验证：不能访问私有数据的节点或者对于那些他们可以访问相关的私有数据但是私有数据是丢失的。如果一个链码的调用既查询又更新私有数据的话，这个提案请求将会返回一个错误。如果你的应用程序能够容忍在链码执行和验证/提交阶段结果集的变动，那么你可以调用一个链码方法来执行这个查询，然后再调用第二个链码方法来执行变更。注意，调用 GetPrivateData() 来获取单独的键值可以跟 PutPrivateData() 调用放在同一个交易中，因为所有的节点都能够基于键版本的哈希来验证键的读取。

在集合中使用索引

使用 CouchDB 作为状态数据库 章节讲解了可以在安装阶段，通过将索引打包在一个 `META-INF/statedb/couchdb/indexes` 的路径下的方式，将索引应用到通道的状态数据库。类似的，也可以通过将索引打包在一个 `META-INF/statedb/couchdb/collections/<collection_name>/indexes` 路径下的方式将索引应用到私有数据集合中。一个索引的实例可以查看 这里。

使用私有数据时的思考

私有数据的删除

Peer 可以周期性地删除私有数据。更多细节请查看上边集合定义属性中的 `blockToLive` 。

另外，重申一下，在提交之前，私有数据存储在 Peer 节点的本地临时数据存储中。这些数据在交易提交之后会自动被删除。但是如果交易没有被提交，私有数据就会一直保存在临时数据存储中。Peer 节点会根据配置文件 `core.yaml` 中的 `peer.gossip.pvtData.transientstoreMaxBlockRetention` 的配置周期性的删除临时存储中的数据。

升级集合定义

要更新集合定义或添加新集合，在链码批准和提交交易中，你可以更新 链码定义并传递新的集合配置例如如果使用 CLI 工具，用 `- collections-config` 标签。 如果在更新链码定义时指定了集合配置，每个现有集合的定义必须包括在内。

更新链码定义时，可以添加新的私有数据集合， 并更新现有的私有数据集合，例如添加新的 成员到现有集合或更改集合定义属性的某一项。 请注意，你不能更新集合名称或 blockToLive（区块活跃）属性，因为不管 peer 的区块高度是多少，持久的 blockToLive 属性是需要的。

当 peer 节点提交具有更新的链码定义的区块时，集合更新生效。请注意，集合不能 被删除，因为在该通道的区块链上可能有不能被删除的先前的私有数据 hash。 无法移除的。

私有数据对账

从 v1.4 开始，加入到已存在的集合中的 Peer 节点在私有数据加入到集合之前，可以自动获取提交到集合的私有数据。

私有数据“对账”也应用在 Peer 节点上，用于确认该接收却未接收到的私有数据，比如由于网络原因没有收到的。以此来追踪在区块提交期间“丢失”的私有数据。

私有数据对账根据 core.yaml 文件中的属性 `peer.gossip.pvtData.reconciliationEnabled` 和 `peer.gossip.pvtData.reconcileSleepInterval` 周期性的发生。Peer 节点会从集合成员节点中定期获取私有数据。

注意私有数据对账特性只适用于 v1.4 以上的 Fabric 节点。