

滴雨科技区块链技术指南

关键内容篇

介绍

Hyperledger Fabric 是分布式账本解决方案的平台，该平台以模块化架构为基础，提供高度的机密性，灵活性，灵活性和可扩展性。它旨在支持不同组件的可插拔实现，并适应整个经济生态系统中存在的复杂性和复杂性。我们建议首次使用的用户从以下介绍的其余部分开始，以熟悉区块链的工作方式以及 Hyperledger Fabric 的特定功能和组件。

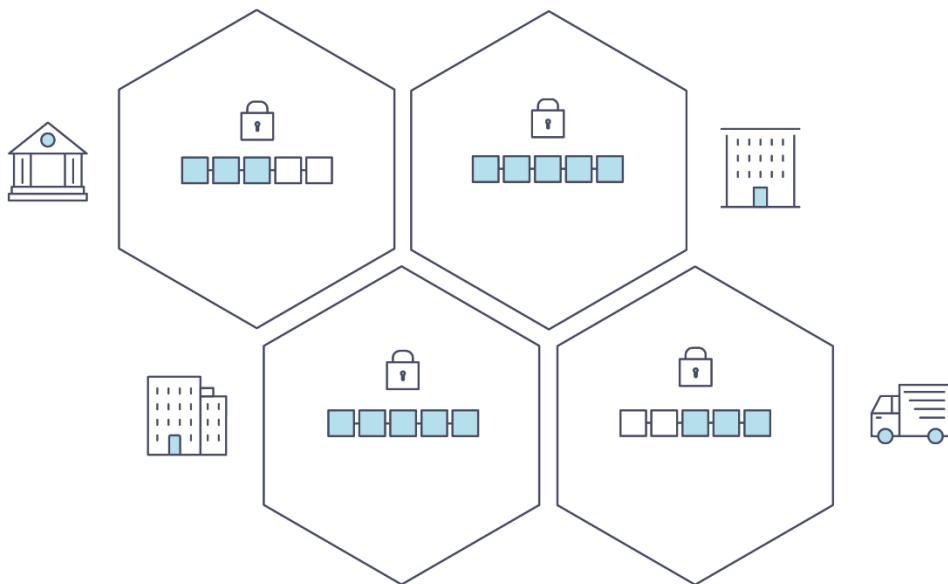
一旦感到舒适（或者如果您已经熟悉区块链和 Hyperledger Fabric），请转到“入门”，然后从中浏览演示，技术规范，API 等。

什么是区块链？

分布式分类帐

区块链网络的核心是分布式账本，记录了网络上发生的所有交易。

区块链分类帐经常被描述为**分散式的或去中心化的**，因为它是在许多网络参与者之间复制的，每个参与者都在维护方面进行**协作**。我们将看到，分散化/去中心化和协作是强大的属性，反映了企业在现实世界中交换商品和服务的方式。



除了去中心化和协作之外，记录到区块链的信息仅是追加的，使用密码学技术来保证一旦将交易添加到分类帐中就无法修改。“不变性”的这种特性使确定信息的来源变得简单，因为参与者可以确定事实之后信息没有被更改。这就是为什么有时将区块链描述为**可证明的系统的原因**。

智能合约

为了支持信息的一致更新-并启用全部分类帐功能（交易，查询等）-区块链网络使用**智能合约**来提供对分类帐的受控访问

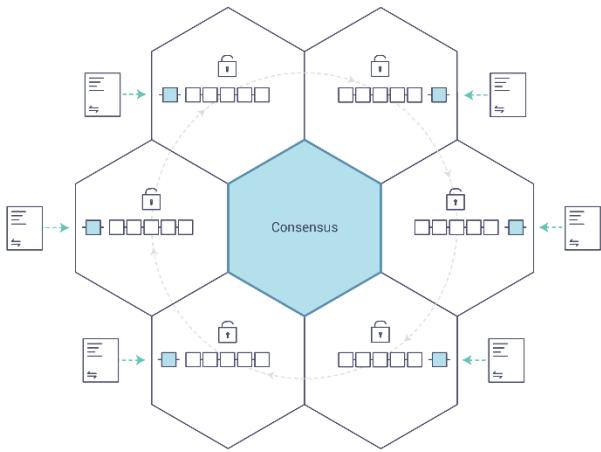


智能合约不仅是封装信息并使之在网络上保持简单的关键机制，还可以编写为允许参与者自动执行交易的某些方面的关键机制。

例如，可以编写智能合约来规定运输物品的成本，其中运输费用取决于物品到达的速度而改变。在双方同意并写入总账的条款下，适当的资金会在收到项目后自动转手。

共识

保持分类账交易在整个网络上同步的过程-确保分类账仅在相应参与者批准交易后更新，并且当分类账确实更新时，它们将以相同的顺序更新相同的交易-称为**共识**



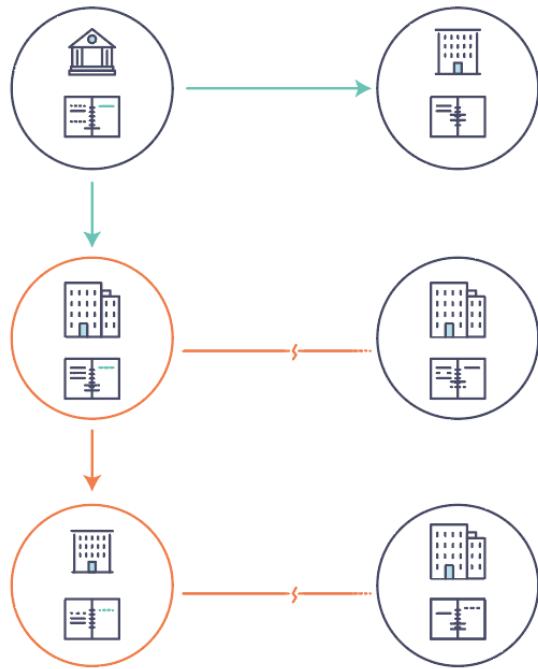
稍后，您将学到更多有关分类账，智能合约和共识的知识。到目前为止，将区块链视为共享的，复制的交易系统就足够了，该系统通过智能合约进行更新，并通过称为共识的协作过程保持一致的同步。

为什么区块链有用？

当前的记录系统

如今的交易网络仅是自保存业务记录以来已经存在的网络的稍微更新版本。**业务网络**的成员彼此进行事务处理，但是它们维护各自的交易记录。他们正在交易的东西-无论是 16 世纪的佛兰德挂毯还是今天的证券-每次出售时都必须确定其出处，以确保出售物品的企业拥有所有权链来验证其所有权。

给您的是一个如下所示的业务网络：



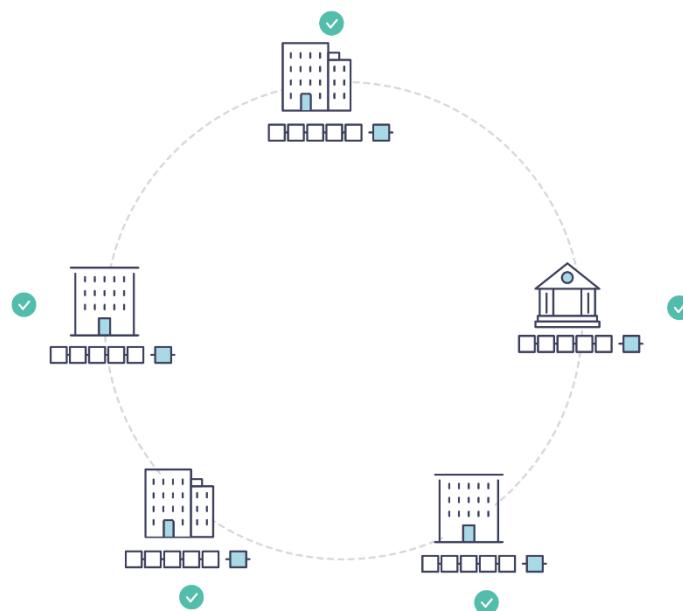
现代技术已将这一过程从石碑和纸质文件夹转移到硬盘驱动器和云平台，但是其基本结构是相同的。尚不存在用于管理网络参与者身份的统一系统，建立来源非常费力，以至于需要数天才能完成证券交易（全球交易量已达数万亿美元），必须手动签署和执行合同，并且系统中的每个数据库都包含唯一信息，因此代表单点故障。

尽管对可见性和信任的需求很明确，但如今采用零散的信息和流程共享方法来建立跨业务网络的记录系统是不可能的。

区块链差异

如果业务网络具有在网络上建立身份，执行事务和存储数据的标准方法，而是由“现代”交易系统所代表的，而不是效率低下的巢穴，那该怎么办？如果可以通过查看一系列交易来确定资产的来源，该交易一经写入便无法更改，因此可以被信任，该怎么办？

该业务网络看起来像这样：



这是一个区块链网络，其中每个参与者都有自己的总账副本。除了共享分类帐信息之外，还共享更新分类帐的过程。与当今的系统（其中参与者的私人程序用于更新其私人分类帐）不同，区块链系统具有共享程序以更新共享的分类帐。

通过共享账本协调业务网络的能力，区块链网络可以减少与私人信息和处理相关的时间，成本和风险，同时提高信任度和可见性。

您现在知道什么是区块链以及为什么有用。还有许多其他重要的细节，但它们都与共享信息和流程的这些基本思想有关。

什么是 Hyperledger Fabric？

Linux 基金会于 2015 年创建了 Hyperledger 项目，以推进跨行业的区块链技术。它没有声明单一的区块链标准，而是鼓励通过社区流程采用协作方法来开发区块链技术，并具有鼓励开放开发和随着时间推移采用关键标准的知识产权。

Hyperledger Fabric 是 Hyperledger 中的区块链项目之一。像其他区块链技术一样，它具有分类帐，使用智能合约，并且是参与者用来管理其交易的系统。

Hyperledger Fabric 与其他一些区块链系统的不同之处在于它是私有的并且被许可。Hyperledger Fabric 网络的成员不是通过允许未知身份参与网络的开放式无许可系统（需要诸如“工作量证明”之类的协议来验证交易并保护网络），而是通过受信任的成员资格服务提供商（MSP）来注册。

Hyperledger Fabric 还提供了几种可插拔选项。账本数据可以以多种格式存储，共识机制可以互换，也可以支持不同的 MSP。

Hyperledger Fabric 还提供了创建渠道的功能，允许一组参与者创建单独的交易分类账。对于某些参与者可能是竞争对手并且不希望他们进行的每笔交易（例如，他们向某些参与者而不是其他参与者提供的特殊价格）的网络，每个参与者都知道这是一个特别重要的选择。如果有两个参与者组成一个渠道，则这些参与者（没有其他参与者）拥有该渠道的分类帐副本。

共享分类帐

Hyperledger Fabric 具有一个账本子系统，该子系统包括两个组件：世界状态和事务日志。每个参与者都有一份账本到他们所属的每个 Hyperledger Fabric 网络的副本。

世界状态组件描述分类帐在给定时间点的状态。这是分类帐的数据库。事务日志组件记录所有导致世界状态当前值的事务；这是世界状态的更新历史。然后，分类帐是世界状态数据库和事务日志历史记录的组合。

分类帐具有世界状态的可替换数据存储。默认情况下，这是一个 LevelDB 键值存储数据库。事务日志不需要是可插入的。它仅记录区块链网络使用的分类帐数据库的前后值。

智能合约

Hyperledger Fabric 智能合约以链码编写，并在区块链外部的应用程序需要与分类账交互时由该应用程序调用。在大多数情况下，chaincode 仅与分类帐的数据库组件，世界状态（例如，查询状态）交互，而不与事务日志交互。

链码可以用几种编程语言实现。当前，支持 Go 和 Node。

隐私

根据网络的需求，企业对企业（B2B）网络的参与者可能对他们共享多少信息非常敏感。对于其他网络，隐私将不是头等大事。

Hyperledger Fabric 支持以隐私（使用通道）为主要操作要求的网络以及相对开放的网络。

共识

即使交易可能在网络内的不同参与者集之间，也必须按照交易发生的顺序将其写入分类帐。为此，必须建立交易顺序，并且必须采用一种方法来拒绝错误（或恶意）插入分类帐中的不良交易。

这是计算机科学领域中经过深入研究的领域，有很多方法可以实现，但都需要权衡取舍。例如，PBFT（实用拜占庭式容错）可以为文件副本提供相互通信的机制，即使在出现损坏的情况下，也可以使每个副本保持一致。或者，在比特币中，排序通过称为挖掘的过程进行，在此过程中，竞争的计算机竞相解决一个密码难题，该难题定义了随后所有进程所依据的顺序。

Hyperledger Fabric 旨在允许网络初学者选择一种共识机制，以最能代表参与者之间存在的关系。与隐私一样，也有各种各样的需求。从关系高度结构化的网络到点对点的网络。

我们将了解有关 Hyperledger Fabric 共识机制的更多信息，该机制目前包括 Kafka 和 Raft。

在哪里可以了解更多？

- [身份](#) (概念性文件)

一个概念性文档，它将带您逐步了解身份在 Fabric 网络中的关键角色（使用已建立的 PKI 结构和 x.509 证书）。

- [成员资格](#) (概念性文档)

通过成员资格服务提供商（MSP）的角色进行对话，后者将身份转换为结构网络中的角色。

- [对等节点](#) (概念性文档)

组织拥有的对等节点托管分类帐和智能合约，并构成 Fabric 网络的物理结构。

- [建立您的第一个网络](#) (BYFN 教程)

了解如何下载 Fabric 二进制文件并使用示例脚本引导您自己的示例网络。然后拆下网络，一次学习如何构建。

- [编写第一个应用程序](#) (教程)

部署一个非常简单的网络（甚至比构建您的第一个网络更简单）以与简单的智能合约和应用程序一起使用。

- [交易流程](#)

从高层次看样本交易流。

- [超级账本结构模型](#)

对本引言中提出的某些组件和概念以及其他一些组件和概念进行了高级介绍，并描述了它们如何在示例事务流中一起工作。

超级账本 Fabric 功能

Hyperledger Fabric 是分布式账本技术(DLT)的实现，它以模块化的区块链架构提供企业就绪的网络安全性，可伸缩性，机密性和性能。Hyperledger Fabric 提供以下区块链网络功能：

身份管理

为了启用许可的网络，Hyperledger Fabric 提供了会员身份服务，该服务管理用户 ID 并验证网络上的所有参与者。访问控制列表可用于通过特定网络操作的授权来提供其他权限层。例如，可以允许特定的用户 ID 调用链码应用程序，但是阻止其部署新的链码。

隐私权和机密性

Hyperledger Fabric 使竞争的商业利益以及任何需要私人机密交易的组能够共存于同一许可的网络上。专用通道是受限制的消息传递路径，可用于为网络成员的特定子集提供事务保密性和机密性。通道上的所有数据（包括事务，成员和通道信息）对于任何未明确授予该通道访问权限的网络成员都是不可见和不可访问的。

高效处理

Hyperledger Fabric 通过节点类型分配网络角色。为了向网络提供并发性和并行性，将事务执行与事务排序和承诺分开。在订购交易之前执行交易使每个对等节点可以同时处理多个交易。这种并发执行提高了每个对等方的处理效率，并加速了向排序服务的交易交付。

除了支持并行处理之外，分工还减轻了排序节点的负担，使其无需执行事务和进行分类帐维护，同时使对等节点免于排序（协商）工作量。角色的这种分叉还限制了授权和身份验证所需的处理。所有对等节点不必信任所有排序节点，反之亦然，因此一个节点上的进程可以独立于另一个节点的验证运行。

链码功能

链码应用程序对逻辑进行编码，该逻辑由通道上特定类型的事务调用。例如，定义用于资产所有权变更的参数的链码可确保所有转让所有权的交易都遵循相同的规则和要求。**系统链码**被区分为定义整个通道的操作参数的链码。生命周期和配置系统链码定义了通道规则；认可和验证系统链码定义了认可和验证交易的要求。

模块化设计

Hyperledger Fabric 实现了模块化架构，可以为网络设计人员提供功能选择。例如，可以将用于身份，排序（共识）和加密的特定算法插入任何 Hyperledger Fabric 网络。结果是任何行业或公共领域均可采用的通用区块链架构，并确保其网络可跨市场，法规和地理边界互操作。

超级账本 Fabric 模型

本节概述了编织到 Hyperledger Fabric 中的关键设计功能，这些功能可以实现其对全面但可自定义的企业区块链解决方案的承诺：

- **资产** -资产定义允许通过网络交换几乎所有具有货币价值的东西，从完整食品到古董车再到货币期货。
- **链码** -链码执行从事务顺序中划分出来，限制了节点类型之间的信任和验证级别，并优化了网络可伸缩性和性能。
- **分类帐功能** -不变的共享分类帐对每个通道的整个交易历史进行编码，并包括类似 SQL 的查询功能，以进行有效的审计和争议解决。
- **隐私** -渠道和私人数据收集可实现私有和机密的多边交易，这通常是竞争企业和受管制的行业（在同一网络上交换资产）进行的。
- **安全和会员服务** -允许的会员提供了一个受信任的区块链网络，参与者知道所有交易都可以由授权的监管机构和审计员检测和追踪。
- **共识** -达成共识的独特方法可实现企业所需的灵活性和可扩展性。

资产

资产的范围从有形的（房地产和硬件）到无形的（合同和知识产权）。Hyperledger Fabric 提供了使用链码交易修改资产的功能。

资产在 Hyperledger Fabric 中表示为键值对的集合，状态更改记录为通道 分类账中的事务。资产可以二进制和/或 JSON 形式表示。

链码

Chaincode 是定义一项或多项资产的软件，以及用于修改资产的交易指令；换句话说，这是业务逻辑。Chaincode 强制执行用于读取或更改键值对或其他状态数据库信息的规则。链码功能针对分类帐的当前状态数据库执行，并通过交易建议启动。链码执行会产生一组键值写操作（写集），这些键值写操作可以提交给网络，并应用于所有对等方的分类帐中。

分类帐功能

分类帐是结构中所有状态转换的有序，防篡改记录。状态转换是参与方提交的链码调用（“交易”）的结果。每笔交易都会产生一组资产键值对，这些键值对在创建，更新或删除时将被提交到分类帐。

分类帐由一个区块链（“chain”）和一个状态数据库组成，该区块链将不可变的顺序记录存储在块中，该状态数据库用于维护当前的结构状态。每个频道有一个分类帐。每个对等方都为其所属的每个通道维护一个分类帐的副本。

Fabric 分类帐的一些功能：

- 使用基于键的查找，范围查询和组合键查询来查询和更新分类帐
- 使用丰富查询语言的只读查询（如果使用 CouchDB 作为状态数据库）
- 只读历史记录查询—查询密钥的分类帐历史记录，从而启用数据出处场景
- 事务包括以链码（读集）读取的键/值的版本和以链码（写集）写入的键/值的版本。
- 交易包含每个背书对等方的签名，并提交给排序服务
- 交易被分为几大块，并从排序服务“交付”到渠道上的对等方
- 对等方根据背书政策验证交易并执行政策
- 在附加块之前，执行版本检查，以确保自链码执行以来，已读取资产的状态未更改
- 一旦交易被确认并提交，就不会有不可改变性
- 通道的分类帐包含一个配置块，用于定义策略，访问控制列表和其他相关信息
- 通道包含成员资格服务提供者实例，允许从不同的证书颁发机构派生加密材料
- 有关数据库，存储结构和“查询能力”的更深入了解，请参阅分类帐主题。

隐私

Hyperledger Fabric 在每个通道的基础上使用不可变的分类帐，以及可以操纵和修改资产当前状态（即更新键值对）的链码。分类账存在于渠道范围内-可以在整个网络中共享（假设每个参与者都在一个公共渠道上工作）-或可以将其私有化以仅包括一组特定的参与者。

在后一种情况下，这些参与者将创建一个单独的渠道，从而隔离/隔离他们的交易和分类帐。为了解决想要弥合总体透明度和隐私之间的差距的方案，只能在需要访问资产状态以执行读写的对等方上安装链码（换句话说，如果未在对等方上安装链码），它将无法与分类帐正确连接）。

当该渠道上的组织子集需要对其交易数据保密时，可以使用私有数据集合（集合）将这些数据隔离在逻辑上与渠道分类帐分离的私有数据库中，该数据库只能由组织的授权子集访问。

因此，渠道使交易对于更广泛的网络而言是不公开的，而集合则对渠道上的组织子集之间的数据保持不公开。

为了进一步模糊数据，可以在将交易发送到排序服务并将块附加到分类帐之前，使用 AES 等通用加密算法对链码中的值进行加密（部分或全部）。一旦加密数据已写入分类帐，则只有拥有用于生成密文的相应密钥的用户才能对其解密。

有关如何在区块链网络上实现隐私的更多详细信息，请参见[私有数据](#)主题。

安全和会员服务

Hyperledger Fabric 支持所有参与者都具有已知身份的交易网络。公钥基础结构用于生成与组织，网络组件以及最终用户或客户端应用程序绑定的加密证书。结果，可以在更广泛的网络和通道级别上操纵和控制数据访问控制。Hyperledger Fabric 的这种“允许”概念，再加上渠道的存在和功能，有助于解决隐私和机密性是最重要的问题。

请参阅成员资格服务提供商（MSP）主题，以更好地了解加密实现以及 Hyperledger Fabric 中使用的签名，验证，验证方法。

共识

在分布式分类帐技术中，共识最近已成为单一功能内特定算法的同义词。但是，共识不仅包括简单地同意交易顺序，而且这种差异在 Hyperledger Fabric 中得到了体现，它在整个交易流程（从提案和认可到订购，验证和承诺）中的基本作用得到了强调。简而言之，共识被定义为对包含一个区块的一组交易的正确性的全面验证。

当区块交易的顺序和结果满足明确的策略标准检查时，最终才能达成共识。这些检查和余额发生在事务的生命周期中，包括使用背书策略来指示哪些特定成员必须背书某个事务类，以及系统链代码以确保这些策略得到执行和维护。在作出承诺之前，对等方将使用这些系统链码来确保保存在足够的认可，并且它们是从适当的实体派生的。此外，在将包含交易的任何块追加到分类账之前，将进行版本控制检查，在此期间将对分类账的当前状态进行同意或同意。

除了进行大量的背书，有效性和版本检查外，还在交易流程的所有方向上进行持续的身份验证。访问控制列表是在网络的分层上实现的（将服务订购到各个通道），有效载荷在交易建议书通过不同的体系结构组件时被重复签名，验证和认证。总而言之，共识不仅限于一批交易的商定顺序。相反，它是一项总体特征，它是交易从提案到承诺的整个过程中不断进行的验证的副产品。

查看[交易流程图](#)以直观表示共识。

区块链网络

本主题将在[概念上](#)描述 Hyperledger Fabric 如何允许组织在形成区块链网络方面进行协作。如果您是架构师，管理员或开发人员，则可以使用本主题深入了解 Hyperledger Fabric 区块链网络中的主要结构和流程组件。本主题将使用一个易于管理的示例，介绍区块链网络中的所有主要组件。在理解了这个示例之后，您可以在文档的其他地方阅读有关这些组件的更多详细信息，或者尝试[构建示例网络](#)。

阅读本主题并了解策略的概念之后，您将对组织为建立控制已部署的 Hyperledger Fabric 网络的策略而需要做出的决策有深入的了解。您还将了解组织如何使用声明性策略（Hyperledger Fabric 的一项关键功能）来管理网络演进。简而言之，您将了解 Hyperledger Fabric 的主要技术组件以及组织需要就它们做出的决策。

什么是区块链网络？

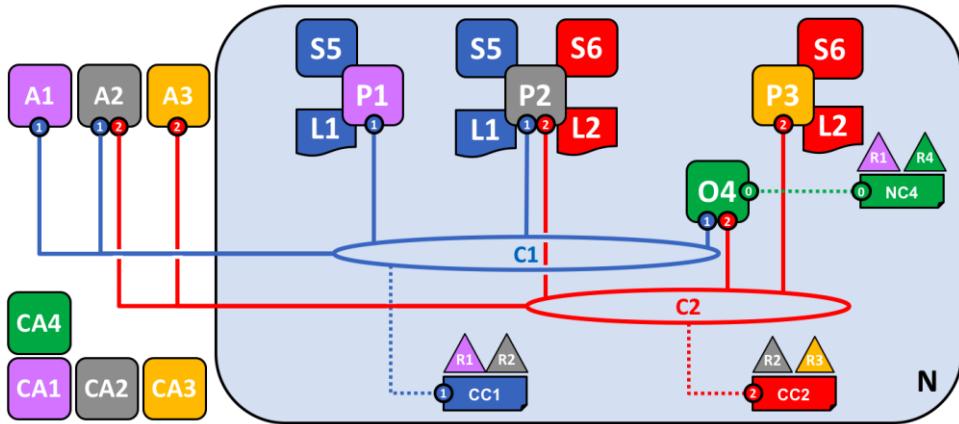
区块链网络是一种技术基础架构，可为应用程序提供分类帐和智能合约（链码）服务。首先，智能合约用于生成交易，随后将交易分配到网络中的每个对等节点，在该对等节点中，它们一成不变地记录在账本副本中。应用程序的用户可能是使用客户端应用程序或区块链网络管理员的最终用户。

在大多数情况下，多个[组织](#)会作为一个[联盟](#)聚集在一起以形成网络，并且它们的权限由最初配置网络时联盟所同意的一组[策略](#)确定。此外，网络策略可能会随着时间的推移而改变，具体取决于联盟内部组织的同意，这在我们讨论[修改策略](#)的概念时会发现。

样本网络

在开始之前，让我们向您展示我们的目标！这是代表示例网络[最终状态](#)的图表。

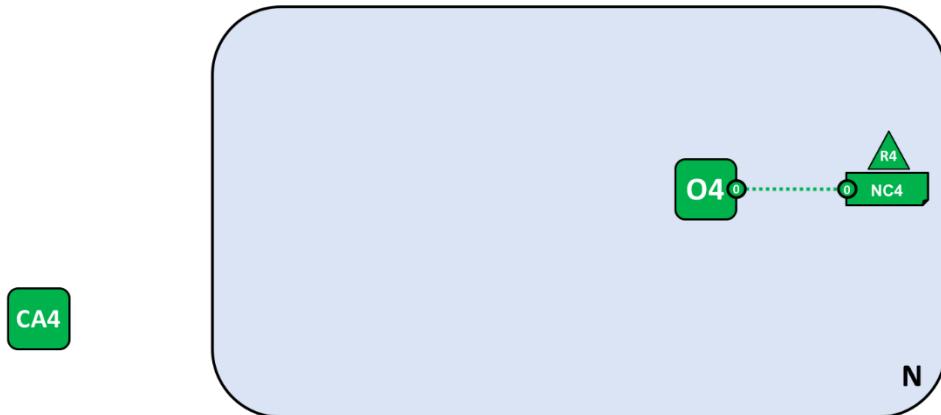
不必担心这看起来很复杂！在研究本主题时，我们将逐步构建网络，以便您了解组织 R1，R2，R3 和 R4 如何为网络贡献基础结构以帮助形成网络。该基础架构实现了区块链网络，并受组成网络的组织（例如，可以添加新组织的组织）所同意的策略的约束。您将发现应用程序如何使用由区块链网络提供的分类帐和智能合约服务。



R1, R2, R3 和 R4 这四个组织共同决定并签署协议，他们将建立和利用 Hyperledger Fabric 网络。R4 已被指定为网络发起者-已被授予建立网络初始版本的权力。R4 无意在网络上执行业务交易。R1 和 R2 以及 R2 和 R3 都需要在整个网络内进行专用通信。组织 R1 有一个客户端应用程序，可以在通道 C1 内执行业务交易。组织 R2 有一个客户端应用程序，可以在通道 C1 和 C2 中完成类似的工作。组织 R3 有一个客户端应用程序，可以在通道 C2 上执行此操作。对等节点 P1 维护与 C1 关联的分类帐 L1 的副本。对等节点 P2 维护与 C1 关联的分类帐 L1 的副本和与 C2 关联的分类帐 L2 的副本。对等节点 P3 维护与 C2 关联的分类帐 L2 的副本。该网络根据网络配置 NC4 中指定的策略规则进行管理，该网络受组织 R1 和 R4 的控制。通道 C1 根据通道配置 CC1 中指定的策略规则进行管理；该渠道受组织 R1 和 R2 的控制。通道 C2 根据通道配置 CC2 中指定的策略规则进行管理；该渠道受组织 R2 和 R3 的控制。有一个排序服务 O4，它充当 N 的网络管理点，并使用系统通道。排序服务还支持应用程序通道 C1 和 C2，出于交易目的，将其分为块进行分发。四个组织中的每一个都有一个首选的证书颁发机构。

建立网络

让我们从创建网络的基础开始：



启动排序者后便形成了网络。在我们的示例网络 N 中，包括单个节点 O4 的排序服务是根据网络配置 NC4 配置的，该网络配置向组织 R4 提供管理权限。在网络级别，证书颁发机构 CA4 用于将身份分配给 R4 组织的管理员和网络节点。我们可以看到，**定义网络的第一件事 N 是排序服务 O4**。将排序服务视为网络的初始管理点很有帮助。按照事先约定，O4 最初由组织 R4 中的管理员配置和启动，并托管在 R4 中。配置 NC4 包含描述网络管理功能的初始集合的策略。最初，将其设置为仅通过网络授予 R4 权限。这将改变，我们将在后面看到，但是目前 R4 是网络的唯一成员。

证书颁发机构

您还可以看到证书颁发机构 CA4，它用于向管理员和网络节点颁发证书。CA4 在我们的网络中起着关键作用，因为它分发 X.509 证书，该证书可用于将组件标识为属于组织 R4。由 CA 颁发的证书还可以用于签署交易，以表明组织认可交易结果-这是将其接受到分类帐中的前提。让我们更详细地研究 CA 的这两个方面。

首先，区块链网络的不同组件使用证书将自己标识为来自特定组织。这就是为什么通常有不止一个 CA 支持一个区块链网络-不同的组织经常使用不同的 CA。我们将在网络中使用四个 CA。每个组织一个。确实，CA 非常重要，因此 Hyperledger Fabric 为您提供了一个内置的 CA (称为 *Fabric-CA*) 来帮助您前进，尽管在实践中，组织将选择使用自己的 CA。

证书到成员组织的映射是通过称为成员资格服务提供程序 (MSP) 的结构实现的。网络配置 NC4 使用命名的 MSP

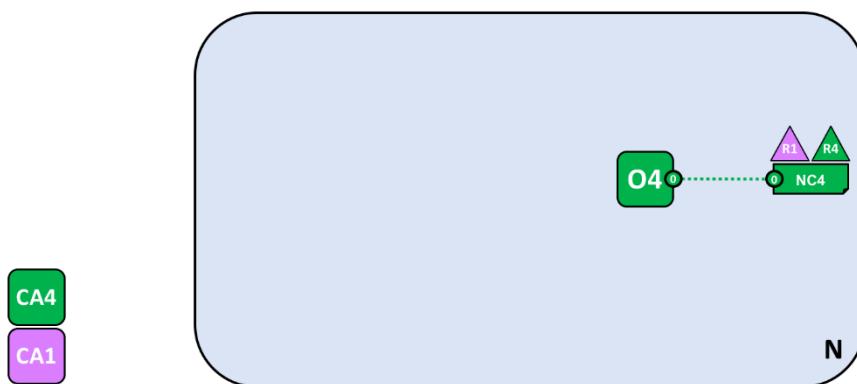
来标识由 CA4 分配的证书的属性，该证书将证书持有者与组织 R4 相关联。然后，NC4 可以在策略中使用此 MSP 名称，以向 R4 的参与者授予对网络资源的特定权限。这种策略的一个示例是确定 R4 中可以将新成员组织添加到网络的管理员。我们不会在这些图中显示 MSP，因为它们只会使它们混乱，但是它们非常重要。

其次，稍后我们将看到 CA 颁发的证书如何成为 事务生成和验证过程的核心。具体而言，X.509 证书用于客户端应用程序 交易建议和智能合约 交易响应中以进行数字签名 交易。随后，托管分类帐副本的网络节点在接受交易到分类帐之前会验证交易签名是否有效。

让我们回顾一下示例区块链网络的基本结构。有一个网络 N 资源，由证书颁发机构 CA4 定义的一组用户访问，这些用户对网络 N 中的资源具有一组权限，如网络配置 NC4 中包含的策略所述。当我们配置并启动排序服务节点 O4 时，所有这些都变为现实。

添加网络管理员

最初将 NC4 配置为仅允许 R4 用户通过网络进行管理。在下一个阶段，我们将允许组织 R1 用户管理网络。让我们看看网络如何发展：



组织 R4 更新网络配置，使组织 R1 也成为管理员。此后，R1 和 R4 在网络配置上享有同等的权利。

我们看到增加了一个新的组织 R1 作为管理员-R1 和 R4 现在网络拥有平等的权利。我们还可以看到已添加了证书颁发机构 CA1 -可用于识别 R1 组织中的用户。此后，R1 和 R4 的用户都可以管理网络。

尽管排序者节点 O4 在 R4 的基础结构上运行，但是 R1 对其具有共享的管理权限，只要它可以获得网络访问权限即可。这意味着 R1 或 R4 可以更新网络配置 NC4，以允许 R2 组织进行网络操作的子集。这样，即使 R4 正在运行排序服务，并且 R1 对其拥有完全的管理权限，R2 仍具有创建新联合体的有限权限。

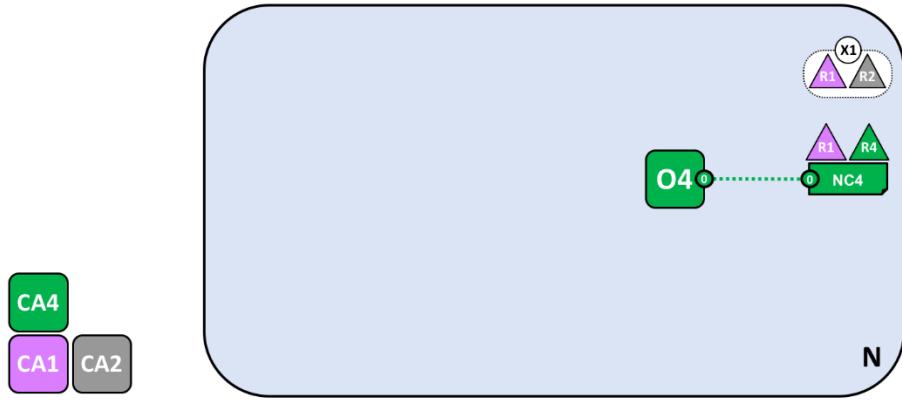
以最简单的形式，排序服务是网络中的单个节点，这就是您在示例中看到的内容。排序服务通常是多节点的，可以配置为在不同组织中具有不同的节点。例如，我们可以在 R4 中运行 O4 并将其连接到组织 R1 中单独的排序节点节点 O2。这样，我们将拥有一个站点，多组织的管理结构。

在本主题的稍后部分，我们将讨论排序服务，但现在，只需将排序服务视为一个管理点，即可为不同组织提供对网络的受控访问。

定义共同体

尽管现在可以通过 R1 和 R4 管理网络，但是几乎没有什么可以完成的。我们需要做的第一件事是定义一个共同体。这个词的字面意思是“一个拥有共同命运的团体”，因此对于区块链网络中的一组组织来说是一个适当的选择。

让我们看看如何定义一个联盟：



网络管理员定义了一个联盟 $X1$ ，该联盟包含两个成员，组织 $R1$ 和 $R2$ 。该联盟定义存储在网络配置 $NC4$ 中，并将在网络开发的下一阶段使用。 $CA1$ 和 $CA2$ 是这些组织各自的证书颁发机构。

由于 $NC4$ 的配置方式，只有 $R1$ 或 $R2$ 可以创建新的联合体。该图显示了添加的新联盟 $X1$ ，该联盟将 $R1$ 和 $R2$ 定义为其组成组织。我们还可以看到已经添加了 $CA2$ 以便 $R2$ 识别用户。请注意，一个联盟可以有任意数量的组织成员—我们刚刚显示了两个，因为它是最简单的配置。

共同体为什么重要？我们可以看到，一个联盟定义了网络中彼此共享交易需求的组织集合—在这种情况下为 $R1$ 和 $R2$ 。如果组织有共同的目标，将它们组合在一起真的很有意义，而这正是正在发生的事情。

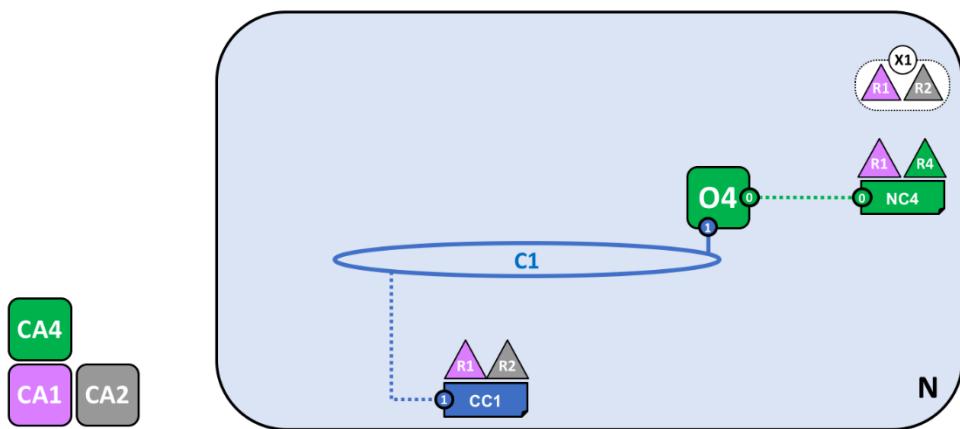
网络虽然由单个组织启动，但现在由更大的一组组织控制。我们可以以这种方式启动它，让 $R1$ 、 $R2$ 和 $R4$ 拥有共享控制权，但是这种积累使它更易于理解。

现在，我们将使用联盟 $X1$ 创建 Hyperledger Fabric 区块链的一个非常重要的部分—**一个通道**。

为联盟或共同体创建渠道

因此，让我们创建 Fabric 区块链网络的关键部分—**一个 channel**。通道是主要的通信机制，联盟的成员可以通过该机制相互通信。网络中可以有多个渠道，但现在，我们将从一个渠道开始。

让我们看看如何将第一个频道添加到网络中：



使用联盟定义 $X1$ 为 $R1$ 和 $R2$ 创建了通道 $C1$ 。通道由完全独立于网络配置的通道配置 $CC1$ 控制。 $CC1$ 由对 $C1$ 拥有同等权利的 $R1$ 和 $R2$ 管理。 $R4$ 在 $CC1$ 中没有任何权利。

通道 $C1$ 为联盟 $X1$ 提供了专用的通信机制。我们可以看到通道 $C1$ 已连接到排序服务 $O4$ ，但是没有附加任何内容。在网络开发的下一阶段，我们将连接组件，例如客户端应用程序和对等节点。但是在这一点上，渠道代表着未来连接的潜力。

即使通道 $C1$ 是网络 N 的一部分，也可以从中区分出来。还要注意，组织 $R3$ 和 $R4$ 不在此通道中—用于 $R1$ 和 $R2$ 之间的事务处理。在上一步中，我们了解了 $R4$ 如何授予 $R1$ 创建新共同体的权限。值得一提的是 $R4$ 还允许 $R1$

创建频道！在此图中，可能是组织 R1 或 R4 创建了通道 C1。同样，请注意，一个通道可以连接任何数量的组织-我们已经展示了两个，因为它是最简单的配置。

再次注意通道 C1 与网络配置 NC4 如何具有完全独立的配置 CC1。CC1 包含用于控制 R1 和 R2 在通道 C1 上拥有的权限的策略-正如我们所看到的，R3 和 R4 在此通道中没有权限。如果 R3 和 R4 由 R1 或 R2 添加到通道配置 CC1 中的适当策略，则 R3 和 R4 只能与 C1 交互。一个示例是定义谁可以向频道添加新组织。特别要注意的是，R4 不能将自己添加到通道 C1 中-它必须并且只能由 R1 或 R2 授权。

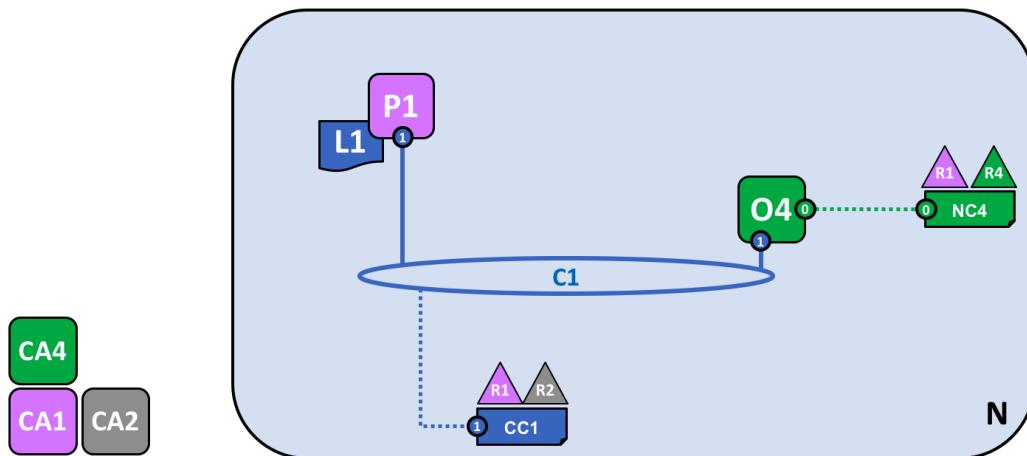
为什么渠道如此重要？通道之所以有用，是因为它们为联盟成员之间的私有通信和私有数据提供了一种机制。频道可提供其他频道和网络的保密性。Hyperledger Fabric 在这方面非常强大，因为它允许组织共享基础结构并同时保持私有状态。这里没有矛盾-网络中的不同联盟将需要适当共享不同的信息和流程，而渠道则提供了一种有效的机制来做到这一点。通道可有效共享基础架构，同时保持数据和通信的隐私。

我们还可以看到，一旦创建了一个频道，它实际上就是“脱离网络”。从此时到将来，只有在通道配置中明确指定的组织才能对其进行控制。同样，从此时开始对网络配置 NC4 的任何更新都不会直接影响通道配置 CC1；例如，如果联盟定义 X1 更改，它将不会影响通道 C1 的成员。通道之所以有用，是因为它们允许构成通道的组织之间进行私人通信。此外，通道中的数据与网络的其余部分（包括其他通道）完全隔离。

顺便说一句，还有一个特殊的**系统通道**，供排序服务使用。它的行为与常规通道完全相同，因此有时会称为**应用程序通道**。通常，我们无需担心该渠道，但是在本主题后面的部分中，我们将对此进行更多讨论。

对等节点和分类帐

现在开始使用通道将区块链网络和组织组件连接在一起。在网络开发的下一阶段，我们可以看到我们的网络 N 刚刚获得了两个新组件，即对等节点 P1 和分类帐实例 L1。



对等节点 P1 已加入通道 C1。P1 物理上存放分类帐 L1 的副本。P1 和 O4 可以使用通道 C1 相互通信。

对等节点是托管区块链分类帐副本的网络组件！最后，我们开始看到一些可识别的区块链组件！P1 在网络中的目的纯粹是托管分类帐 L1 的副本，以供其他人访问。我们可以认为 L1 物理上托管在 P1 上，但逻辑上托管在通道 C1 上。当我们向频道添加更多对等方时，我们会更清楚地看到这个想法。

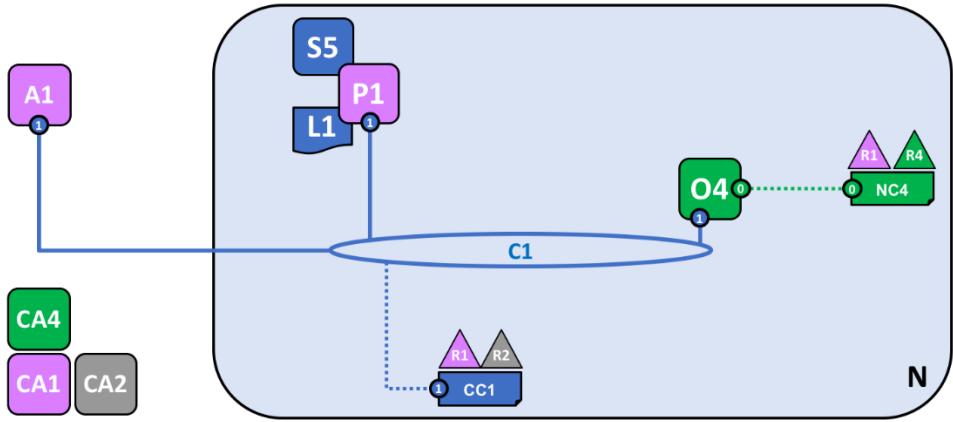
P1 的配置的关键部分是 CA1 发出的 X.509 身份，该身份将 P1 与组织 R1 相关联。一旦 P1 启动，它就可以使用命令程序 O4 加入通道 C1。O4 收到此加入请求时，将使用通道配置 CC1 来确定 P1 在此通道上的权限。例如，CC1 确定 P1 是否可以向分类帐 L1 读取和/或写入信息。

请注意，对等点是如何由拥有它们的组织加入渠道的，尽管我们仅添加了一个对等点，但我们将看到网络中多个渠道上如何有多个对等节点。我们将在稍后看到对等节点可以扮演的不同角色。

应用程序和智能合约链代码

现在，通道 C1 上有一个分类帐，我们可以开始连接客户端应用程序，以使用分类帐的主要对象（对等方）提供的某些服务！

请注意网络的增长方式：



智能合约 $S5$ 已安装到 $P1$ 上。组织 $R1$ 中的客户端应用程序 $A1$ 可以使用 $S5$ 通过对等节点 $P1$ 访问分类帐。 $A1$, $P1$ 和 $O4$ 都已加入通道 $C1$, 即它们都可以利用该通道提供的通信设施。

在网络开发的下一阶段, 我们可以看到客户端应用程序 $A1$ 可以使用通道 $C1$ 连接到特定的网络资源-在这种情况下, $A1$ 可以同时连接到对等节点 $P1$ 和排序节点节点 $O4$ 。再次, 查看渠道如何在网络和组织组件之间进行通信。就像对等方和排序方一样, 客户端应用程序将具有将其与组织关联的身份。在我们的示例中, 客户端应用程序 $A1$ 与组织 $R1$ 关联; 尽管它在 Fabric 区块链网络之外, 但仍通过通道 $C1$ 连接到它。

现在看来, $A1$ 可以直接通过 $P1$ 访问分类帐 $L1$, 但实际上, 所有访问都是通过称为智能合约链码 $S5$ 的特殊程序进行管理的。将 $S5$ 定义为分类帐的所有常见访问模式; $S5$ 提供了一套明确定义的方式, 可以查询或更新分类帐 $L1$ 。简而言之, 客户应用程序 $A1$ 必须通过智能合约 $S5$ 才能到达分类帐 $L1$!

可以由每个组织中的应用开发人员创建智能合约, 以实施联盟成员共享的业务流程。智能合约用于帮助生成交易, 随后可以将交易分配到网络中的每个节点。我们稍后再讨论这个想法。当网络更大时, 将更容易理解。现在, 需要了解的重要一点是, 到此为止, 必须在智能合约上执行两次操作。它必须已安装在对等方上, 然后 在通道上定义。

Hyperledgerfabric 用户经常使用的术语**智能合同**和 **chaincode** 互换。通常, 智能合约定义了控制世界状态中包含的业务对象生命周期的 **交易逻辑**。然后将其打包成链码, 然后将其部署到区块链网络。可以将智能合约视为支配事务, 而链码则可以控制如何打包智能合约以进行部署。

安装 chaincode 包

开发了智能合约 $S5$ 之后, 组织 $R1$ 中的管理员必须创建一个链码包并将其安装到对等节点 $P1$ 上。这是一个简单的操作; 一旦完成, $P1$ 就完全了解 $S5$ 。具体来说, $P1$ 可以看到 $S5$ 的实现逻辑-它用于访问分类帐 $L1$ 的程序代码。我们将其与仅描述 $S5$ 的输入和输出而不考虑其实现的 $S5$ 接口进行对比。

当组织在一个通道中有多个对等方时, 它可以选择在其上安装智能合约的对等方。它不需要在每个对等节点上安装智能合约。

定义链码

尽管链码已安装在各个组织的对等方上, 但它是在渠道范围内管理和操作的。每个组织都需要批准链码定义, 这是一组参数, 用于确定如何在渠道上使用链码。组织必须批准链码定义, 才能使用已安装的智能合约来查询分类账并认可交易。在我们的示例中, 该示例只有一个对等节点 $P1$, 组织 $R1$ 中的管理员必须批准 $S5$ 的链码定义。

足够数量的组织需要批准链码定义(默认情况下为大多数, 大多数情况下), 然后才能将链码定义提交给渠道并用于与渠道分类帐进行交互。由于通道只有一个成员, 因此 $R1$ 的管理员可以将 $S5$ 的链码定义提交给通道 $C1$ 。提交定义后, 现在可以由客户端应用程序 $A1$ 调用 $S5$!

请注意, 尽管道上的每个组件现在都可以访问 $S5$, 但它们无法看到其程序逻辑。对于已安装它的那些节点, 它仍然是私有的; 在我们的示例中, 这意味着 $P1$ 。从概念上讲, 这意味着已定义并提交给通道的是智能合约接口, 与已安装

的智能合约实现相反。加强这个想法；安装智能合约显示了我们如何将其视为 **物理托管** 在对等方上，而已在通道上定义的智能合约则显示了我们如何将其视为**逻辑上**由通道托管。

背书政策

链码定义中提供的最重要的信息是背书策略。它描述了哪些组织必须批准交易，其他组织才能将其接受到其账本副本上。在我们的示例网络中，只有 R1 或 R2 认可交易，才可以将交易接受到分类帐 L1 上。

将链码定义提交给渠道会将背书策略放置在渠道分类帐上。它使通道的任何成员都可以访问它。您可以在[事务流主题](#)中阅读有关背书策略的更多信息。

调用智能合约

将智能合约安装在对等节点上并在通道上定义后，即可由客户端应用程序调用。客户端应用程序通过向智能合约认可策略指定的组织所拥有的对等方发送交易建议来完成此任务。交易建议用作智能合约的输入，智能合约使用它来生成认可的交易响应，该响应由对等节点返回到客户端应用程序。

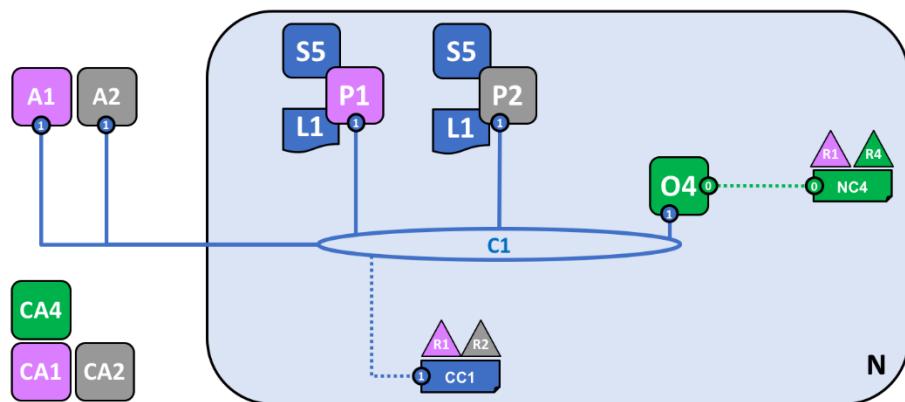
这些交易响应与交易提议一起打包在一起，形成了一个完全认可的交易，可以分布到整个网络。稍后我们将更详细地介绍这一点。现在，足以了解应用程序如何调用智能合约来生成认可的交易。

在网络开发的这一阶段，我们可以看到组织 R1 正在完全参与网络。它的应用程序-从 A1 开始-可以通过智能合约 S5 访问分类帐 L1，以生成将由 R1 认可的交易，因此由于它们符合认可策略而被接受到分类帐中。

网络完成

回想一下，我们的目标是为 X1 联盟（组织 R1 和 R2）创建一个渠道。网络开发的下一阶段将看到组织 R2 将其基础结构添加到网络中。

让我们看看网络是如何发展的：



通过增加组织 R2 的基础结构，网络得到了发展。具体来说，R2 添加了对等节点 P2，该对等节点托管账本 L1 的副本和链码 S5。R2 批准与 R1 相同的链码定义。P2 也与应用程序 A2 一样加入了通道 C1。A2 和 P2 使用来自 CA2 的证书进行标识。所有这些意味着应用程序 A1 和 A2 都可以使用对等节点 P1 或 P2 在 C1 上调用 S5。

我们可以看到组织 R2 在通道 C1 上添加了一个对等节点 P2。P2 还托管分类帐 L1 和智能合约 S5 的副本。我们可以看到 R2 还添加了可以通过通道 C1 连接到网络的客户端应用程序 A2。为此，组织 R2 中的管理员已创建对等节点 P2，并将其加入通道 C1，方法与 R1 中的管理员相同。管理员还必须批准与 R1 相同的链码定义。

我们已经建立了第一个运营网络！在网络开发的现阶段，我们拥有一个渠道，组织 R1 和 R2 可以彼此完全进行交易。具体来说，这意味着应用程序 A1 和 A2 可以使用智能合约 S5 和分类帐 L1 在通道 C1 上生成交易。

生成和接受交易

与始终承载分类帐副本的对等节点相反，我们看到有两种不同的对等节点：那些拥有智能合约的人和没有智能

合约的人。在我们的网络中，每个对等方都承载智能合约的副本，但是在较大的网络中，将有更多的对等节点不承载智能合约的副本。对等方只能在安装了智能合约的情况下运行它，但是通过连接到通道，它可以知道智能合约的接口。

您不应该将没有安装智能合约的对等节点视为劣等节点。具有智能合约的对等节点更具有特殊的功能—帮助生成交易。注意，所有对等节点都可以验证并随后接受或拒绝交易到其账本 L1 的副本上。但是，只有安装了智能合约的对等节点才能参与交易背书的过程，这对生成有效交易至关重要。

我们无需担心本主题中如何生成，分发和接受交易的确切细节，足以了解我们拥有一个区块链网络，组织 R1 和 R2 可以将这些信息和流程共享为分类账捕获的交易。在其他主题中，我们将学习有关交易，分类账，智能合约的更多信息。

对等节点的类型

在 Hyperledger Fabric 中，尽管所有对等体都相同，但是它们可以根据网络的配置方式承担多个角色。现在，我们对典型的网络拓扑有了足够的了解，可以描述这些角色。

- **承诺对等节点**。通道中的每个对等节点都是提交对等节点。它接收生成的交易的块，随后将这些交易进行验证，然后再将它们提交到对等节点的分类帐副本作为追加操作。
- **背书对等节点**。如果安装了智能合约，则拥有智能合约的每个对等方都可以成为背书方。但是，实际上要成为背书方，客户端应用程序必须使用对等方上的智能合约来生成数字签名的事务响应。**背书对等**一词是对此事实的明确引用。

智能合约的签注策略标识了组织，其对等方应先对生成的交易进行数字签名，然后才能将其接受到提交的对等方的分类账副本中。

这是对等体的两种主要类型。对等方可以采用其他两个角色：

- **领导对等节点**。当组织在一个通道中有多个对等方时，领导对等方是负责将交易从排序节点分发到组织中其他提交对等方的节点。对等节点可以选择参与静态或动态领导选择。

因此，从领导者的角度考虑两组对等节点是有帮助的-那些具有静态领导者选择的对等节点和那些具有动态领导者选择的对等节点。对于静态集，可以将零个或多个对等方配置为领导者。对于动态集，将由该集选出一个对等方作为领导者。此外，在动态集中，如果领导者对等方失败，则其余对等方将重新选举领导者。

这意味着组织的对等方可以让一个或多个领导者连接到排序服务。这可以帮助提高处理大量事务的大型网络的弹性和可伸缩性。

- **锚点对等节点**。如果对等方需要与另一个组织中的对等方进行通信，则可以使用该组织的通道配置中定义的**锚点对等方之一**。一个组织可以为其定义零个或多个锚点，并且锚点可以帮助处理许多不同的跨组织通信场景。

注意，一个对等体可以同时是一个提交对等体，认可对等体，领导对等体和锚定对等体！仅锚点对等方是可选的-出于所有实际目的，总会有一个领导方对等方，至少一个背书对等方和至少一个提交对等方。

将组织和对等方添加到渠道

当 R2 加入通道时，组织必须将智能合约 S5 安装到其对等节点 P2 上。这很明显-如果应用程序 A1 或 A2 希望在对等节点 P2 上使用 S5 来生成事务，则必须首先将其存在。安装是发生这种情况的机制。此时，对等节点 P2 具有智能合约和分类帐的物理副本；像 P1 一样，它可以在其分类帐 L1 的副本上生成并接受交易。

为了使用智能合约 S5，R2 必须批准与 R1 批准的相同的链码定义。由于链码定义已由组织 R1 提交给通道，因此，只要组织批准链码定义并安装链码软件包，R2 就可以使用链码。提交事务只需发生一次。新组织批准通道其他成员同意的链码参数后，便可以使用链码。由于链码定义的批准是在组织级别进行的，因此 R2 可以批准链码定义一次，并将安装了链码包的多个对等方加入到通道中。但是，如果 R2 想要更改链码定义，则 R1 和 R2 都需要为其组织批准新的定义，

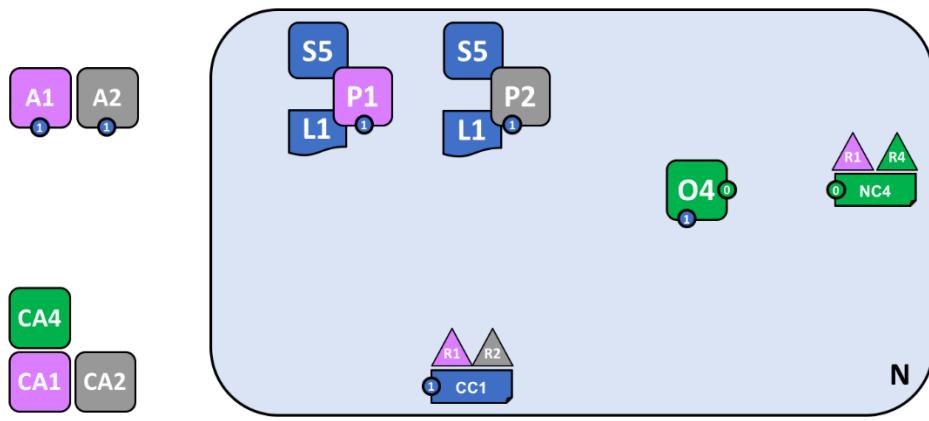
在我们的网络中，我们可以看到通道 C1 连接了两个客户端应用程序，两个对等节点和一个排序服务。由于只

有一个通道，因此只有一个逻辑分类帐可以与这些组件进行交互。对等节点 P1 和 P2 具有分类帐 L1 的相同副本。智能合约 S5 的副本通常将使用相同的编程语言完全相同地实现，但如果不同，则它们在语义上必须等效。

我们可以看到，将对等节点小心地添加到网络可以帮助支持增加的吞吐量，稳定性和弹性。例如，网络中更多的对等点将允许更多的应用程序连接到它；如果计划内或计划外的停机，组织中的多个对等方将提供额外的弹性。这一切都意味着可以配置支持各种操作目标的复杂拓扑—网络可以达到的规模没有理论上的限制。此外，单个组织内的对等方有效地发现并彼此通信的技术机制（gossip 八卦协议）将容纳大量对等节点，以支持此类拓扑。仔细使用网络和通道策略可以使大型网络得到良好管理。组织可以自由地将对等节点添加到网络，只要它们符合网络约定的策略即可。网络和渠道策略在自治和控制之间建立了平衡，这是分散网络的特征。

简化视觉词汇

现在，我们将简化用于表示示例区块链网络的视觉词汇。随着网络规模的扩大，最初用于帮助我们了解渠道的线路将变得很繁琐。想象一下，如果添加另一个对等节点或客户端应用程序或另一个通道，图将变得多么复杂？这就是我们将在一分钟内要做的事情，因此在我们这样做之前，让我们简化视觉词汇。这是到目前为止我们开发的网络的简化表示：



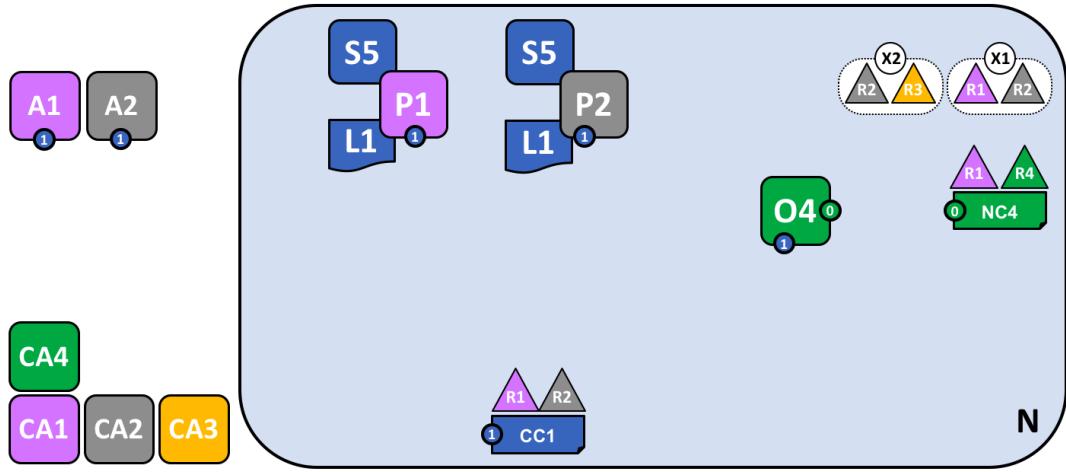
该图显示了与网络 N 中的通道 C1 有关的事实，如下所示：客户端应用程序 A1 和 A2 可以使用通道 C1 与对等端 P1 和 P2 以及排序节点 O4 进行通信。对等节点 P1 和 P2 可以使用通道 C1 的通信服务。排序服务 O4 可以利用通道 C1 的通信服务。通道配置 CC1 适用于通道 C1。

请注意，通过用连接点替换通道线来简化网络图，显示为蓝色圆圈，其中包括通道号。没有信息丢失。此表示形式更具伸缩性，因为它消除了交叉线。这使我们可以更清楚地表示更大的网络。通过专注于组件和通道之间的连接点，而不是通道本身，我们已经实现了这种简化。

添加另一个联盟定义

在网络开发的下一阶段，我们介绍组织 R3。我们将为组织 R2 和 R3 提供一个单独的应用程序通道，使它们能够彼此进行交易。该应用程序通道将与先前定义的通道完全分开，因此 R2 和 R3 事务可以保持私有。

让我们回到网络级别，为 R2 和 R3 定义一个新的联盟 X2：



组织 $R1$ 或 $R4$ 的网络管理员添加了新的联盟定义 $X2$ ，其中包括组织 $R2$ 和 $R3$ 。这将用于为 $X2$ 定义一个新通道。

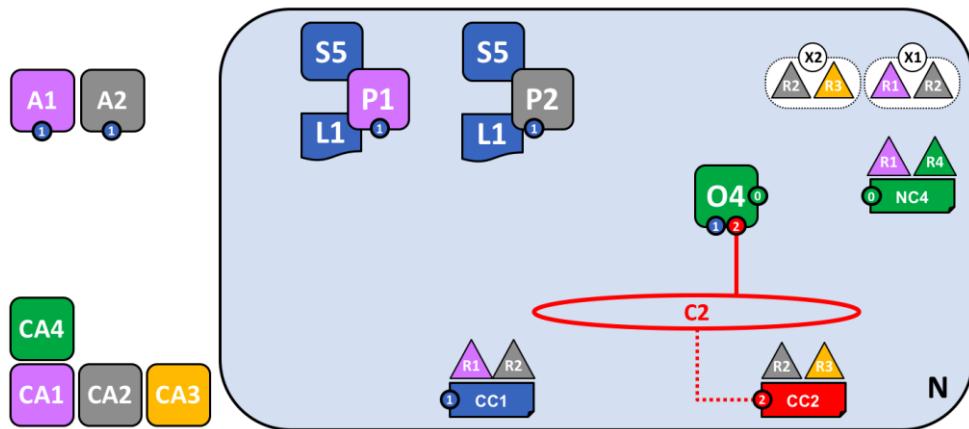
请注意，网络现在定义了两个联盟：组织 $R1$ 和 $R2$ 使用 $X1$ ，组织 $R2$ 和 $R3$ 使用 $X2$ 。引入了联盟 $X2$ ，以便能够为 $R2$ 和 $R3$ 创建新的通道。

只有在网络配置策略 $NC4$ 中明确标识为具有相应权限的组织（即 $R1$ 或 $R4$ ）才能创建新通道。这是策略的示例，该策略将可以在网络级别管理资源的组织与可以在通道级别管理资源的组织区分开来。看到这些策略在起作用，有助于我们理解为什么 Hyperledger Fabric 具有复杂的分层策略结构。

实际上，联盟定义 $X2$ 已添加到网络配置 $NC4$ 中。我们将在文档的其他地方讨论此操作的确切机制。

新增频道

现在，让我们使用这个新的联盟定义 $X2$ 创建一个新的渠道 $C2$ 。为了帮助您进一步了解更简单的通道符号，我们使用了两种视觉样式-通道 $C1$ 用蓝色圆形端点表示，而通道 $C2$ 用红色连接线表示：



使用联盟定义 $X2$ 为 $R2$ 和 $R3$ 创建了一个新的通道 $C2$ 。该通道具有完全独立于网络配置 $NC4$ 的通道配置 $CC2$ 和通道配置 $CC1$ 。通道 $C2$ 由 $R2$ 和 $R3$ 管理，它们具有 $CC2$ 中策略定义的对 $C2$ 相同的权限。 $R1$ 和 $R4$ 在 $CC2$ 中均未定义任何权限。

通道 $C2$ 为联盟 $X2$ 提供了专用的通信机制。同样，请注意组织如何联合起来是什么形式的渠道。通道配置 $CC2$ 现在包含管理通道资源的策略，并通过通道 $C2$ 向组织 $R2$ 和 $R3$ 分配管理权限。它仅由 $R2$ 和 $R3$ 管理； $R1$ 和 $R4$ 在通道 $C2$ 中没有权限。例如，随后可以更新通道配置 $CC2$ 以添加组织来支持网络增长，但这只能由 $R2$ 或 $R3$ 完成。

请注意，通道配置 $CC1$ 和 $CC2$ 如何保持彼此完全独立，并与网络配置 $NC4$ 完全独立。再次，我们看到了

Hyperledger Fabric 网络的去中心化性质。创建通道 C2 后，组织 R2 和 R3 会独立于其他网络元素来管理它。渠道策略始终保持彼此独立，并且只能由有权在渠道中这样做的组织进行更改。

随着网络和通道的发展，网络和通道的配置也将随之发展。有一个过程可以通过受控的方式完成此过程-涉及配置事务，这些事务捕获对这些配置的更改。每次配置更改都会导致生成新的配置块事务，在本主题的后面，我们将看到如何验证和接受这些块以分别创建更新的网络和通道配置。

网络和渠道配置

在整个示例网络中，我们看到了网络和通道配置的重要性。这些配置很重要，因为它们封装了网络成员同意的 **策略**，这些**策略**为控制对网络资源的访问提供了共享的参考。网络和通道配置还包含有关网络和通道组成的事**实**，例如联盟名称及其组织。

例如，当首先使用排序服务节点 O4 形成网络时，其行为由网络配置 NC4 控制。NC4 的初始配置仅包含允许组织 R4 管理网络资源的策略。随后将 NC4 更新为还允许 R1 管理网络资源。进行此更改后，组织 R1 或 R4 中连接到 O4 的任何管理员都将具有网络管理权限，因为这是 NC4 网络配置中允许的策略。在内部，排序服务中的每个节点都会记录网络配置中的每个通道，以便在网络级别上记录每个创建的通道。

这意味着尽管排序服务节点 O4 是创建联盟 X1 和 X2 以及通道 C1 和 C2 的**参与者**，但网络的**智能**包含在 O4 遵循的网络配置 NC4 中。只要 O4 表现良好，并且在处理网络资源时正确执行 NC4 中定义的策略，我们的网络就会按照所有组织的同意行事。在许多方面，NC4 比 O4 更重要，因为它最终控制了网络访问。

相同的原则适用于有关对等方的通道配置。在我们的网络中，P1 和 P2 同样是好演员。当对等节点 P1 和 P2 与客户端应用程序 A1 或 A2 交互时，它们每个都使用在通道配置 CC1 中定义的策略来控制对通道 C1 资源的访问。

例如，如果 A1 要访问对等节点 P1 或 P2 上的智能合约链代码 S5，则每个对等节点都使用其 CC1 副本确定 A1 可以执行的操作。例如，可以根据 CC1 中定义的策略允许 A1 从分类账 L1 读取或写入数据。稍后我们将看到通道及其通道配置 CC2 中参与者的相同模式。再次，我们可以看到，尽管对等方和应用程序是网络中的关键角色，但它们在通道中的行为更多地由通道配置策略决定，而不是其他任何因素。

最后，了解物理上如何实现网络和通道配置将很有帮助。我们可以看到网络和通道配置在逻辑上是唯一的-网络有一个，每个信道有一个。这个很重要；访问网络或通道的每个组件都必须对授予不同组织的权限有共同的了解。

即使从逻辑上讲只有一个配置，但实际上它被构成网络或通道的每个节点复制并保持一致。例如，在我们的网络中，对等节点 P1 和 P2 都具有通道配置 CC1 的副本，而到网络完全完成时，对等节点 P2 和 P3 都将具有通道配置 CC2 的副本。类似地，排序服务节点 O4 具有网络配置的副本，但是在**多节点配置**中，每个排序服务节点将具有其自己的网络配置副本。

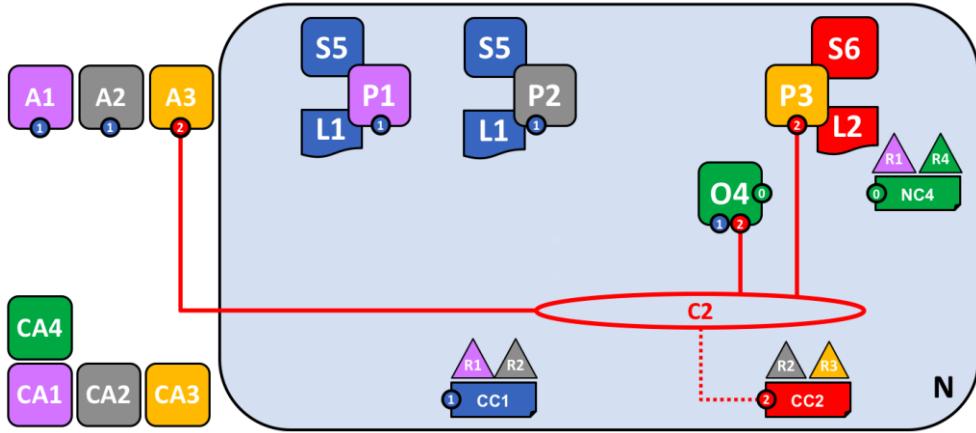
使用用于用户交易（但用于**配置交易**）的相同区块链技术，可以使网络和通道配置保持一致。要更改网络或通道配置，管理员必须提交配置事务以更改网络或通道配置。必须由适当策略中确定负责配置更改的组织签名。该策略称为 **mod_policy**，我们将在后面讨论。

的确，排序服务节点运行着一个微型区块链，通过我们前面提到的**系统通道连接**。使用系统通道排序服务节点可以分发网络配置事务。这些事务用于在每个排序服务节点上合作维护网络配置的一致副本。以类似的方式，**应用程序通道**中的对等节点可以分发通道配置事务。同样，这些事务用于在每个对等节点上维护通道配置的一致副本。通过物理分布在逻辑上是奇异的对象之间的这种平衡是 Hyperledger Fabric 中的常见模式。例如，逻辑上单一的对象（如网络配置）实际上是在一组排序服务节点之间进行物理复制的。我们还会在通道配置，分类帐以及某种程度上将智能合约安装在多个位置，但它们的接口逻辑上存在于通道级别的情况下看到它。您会在 Hyperledger Fabric 中一次又一次地看到这种模式，并使 Hyperledger Fabric 既分散又可管理。

添加另一个对等节点

既然组织 R3 能够完全参与渠道 C2，那么让我们将其基础结构组件添加到渠道中。我们不会一次添加一个组件，而是一次添加一个对等方，其分类帐的本地副本，智能合约和客户端应用程序！

让我们看一下添加了组织 R3 组件的网络：



该图显示了与网络 N 中的通道 $C1$ 和 $C2$ 有关的事实，如下所示：客户端应用程序 $A1$ 和 $A2$ 可以使用通道 $C1$ 与对等端 $P1$ 和 $P2$ 进行通信，并排序服务 $O4$ 。客户端应用程序 $A3$ 可以使用通道 $C2$ 与对等端 $P3$ 和排序服务 $O4$ 进行通信。排序服务 $O4$ 可以利用通道 $C1$ 和 $C2$ 的通信服务。通道配置 $CC1$ 适用于通道 $C1$ ， $CC2$ 适用于通道 $C2$ 。

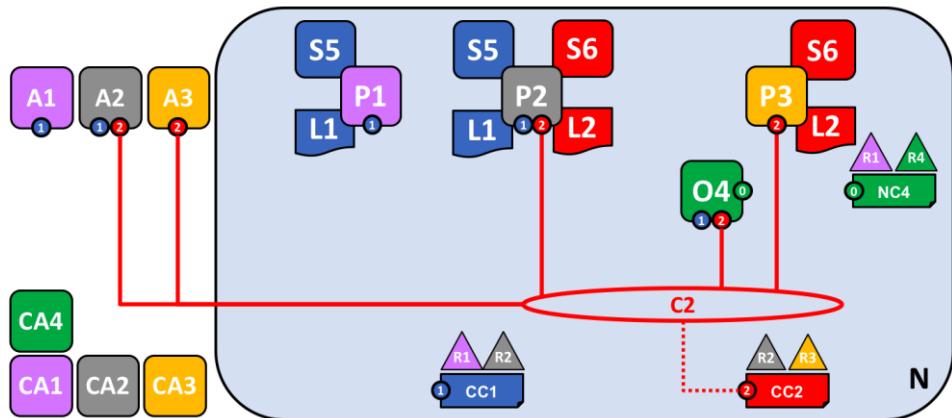
首先，请注意，由于对等节点 $P3$ 连接到通道 $C2$ ，因此它对使用通道 $C1$ 的那些对等节点具有**不同的分类帐** $L2$ 。分类帐 $L2$ 有效地限定在通道 $C2$ 上。分类帐 $L1$ 是完全独立的；它的作用域是通道 $C1$ 。这是有道理的—渠道 $C2$ 的目的是在联盟 $X2$ 的成员之间提供私人通信，而分类帐 $L2$ 是其交易的私人商店。

以类似的方式，安装在对等节点 $P3$ 上并在通道 $C2$ 上定义的智能合约 $S6$ 用于提供对分类帐 $L2$ 的受控访问。应用程序 $A3$ 现在可以使用通道 $C2$ 调用由智能合约 $S6$ 提供的服务，以生成可以接受到网络中分类帐 $L2$ 的每个副本上的交易。

此时，我们有了一个网络，其中定义了两个完全独立的通道。这些渠道为组织之间的相互交易提供了独立管理的设施。同样，这是工作中的分权；我们在控制和自治之间取得平衡。这是通过将策略应用到受不同组织控制并影响不同组织的渠道来实现的。

将对等节点加入多个渠道

在网络开发的最后阶段，让我们将重点返回到组织 $R2$ 。通过将 $R2$ 加入多个渠道，我们可以利用 $R2$ 是 $X1$ 和 $X2$ 联盟成员的事实：



该图显示了与网络 N 中的通道 $C1$ 和 $C2$ 有关的事实，如下所示：客户端应用程序 $A1$ 可以使用通道 $C1$ 与对等端 $P1$ 和 $P2$ 进行通信，并排序服务 $O4$ 。客户端应用程序 $A2$ 可以使用通道 $C1$ 与对等端 $P1$ 和 $P2$ 进行通信，并使用通道 $C2$ 与对等端 $P2$ 和 $P3$ 进行通信以及排序服务 $O4$ 。客户端应用程序 $A3$ 可以使用通道 $C2$ 与对等端 $P3$ 和 $P2$ 进行通信。

以及排序服务 O4 进行通信。排序服务 O4 可以利用通道 C1 和 C2 的通信服务。通道配置 CC1 适用于通道 C1，CC2 适用于通道 C2。

我们可以看到 R2 是网络中的一个特殊组织，因为它是两个应用程序通道中唯一的组织！它能够与通道 C1 上的组织 R1 进行交易，同时它还可以与其他通道 C2 上的组织 R3 进行交易。

请注意，对等节点 P2 如何为通道 C1 安装了智能合约 S5，并为通道 C2 安装了智能合约 S6。对等节点 P2 通过不同账本的不同智能合约同时是两个渠道的正式成员。

这是一个非常强大的概念-渠道既提供了组织分离的机制，又提供了组织之间协作的机制。一直以来，此基础结构是由一组独立的组织提供并在它们之间共享的。

同样重要的是要注意，对等节点 P2 的行为受其进行交易的通道的控制非常不同。具体来说，通道配置 CC1 中包含的策略指示了 P2 在通道 C1 中进行事务处理时可用于 P2 的操作，而通道配置 CC2 中的策略则控制了 P2 在通道 C2 中的行为。

同样，这是理想的- R2 和 R1 同意了通道 C1 的规则，而 R2 和 R3 同意了通道 C2 的规则。这些规则是在各自的渠道策略中捕获的-渠道中的每个组件都可以并且必须使用它们来强制执行正确的行为，这已经达成共识。

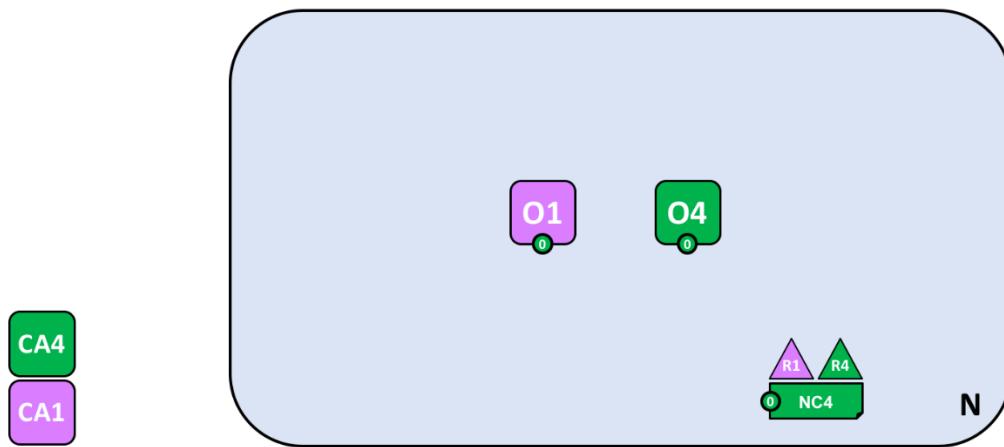
同样，我们可以看到客户端应用程序 A2 现在能够在通道 C1 和 C2 上进行交易。同样，它也将由适当通道配置中的策略控制。顺便说一句，请注意客户端应用程序 A2 和对等节点 P2 正在使用混合的可视词汇表-线路和连接。您可以看到它们是等效的。它们是视觉同义词。

排序服务

细心的读者可能会注意到排序服务节点似乎是一个集中式组件。它最初用于创建网络，并连接到网络中的每个通道。即使我们将 R1 和 R4 添加到控制排序节点的网络配置策略 NC4 中，该节点仍在 R4 的基础结构上运行。在去中心化的世界里，这看起来是错误的！

不用担心 我们的示例网络显示了最简单的排序服务配置，以帮助您了解网络管理点的概念。实际上，排序服务本身也可以完全分散！前面我们提到过，排序服务可以由不同组织拥有的许多单个节点组成，因此让我们看看如何在示例网络中完成该工作。

让我们看一个更现实的排序服务节点配置：



多种组织的排序服务。排序服务包括排序服务节点 O1 和 O4。O1 由组织 R1 提供，节点 O4 由组织 R4 提供。网络配置 NC4 为组织 R1 和 R4 的参与者定义了网络资源权限。

我们可以看到，该排序服务已完全去中心化-在组织 R1 中运行，在组织 R4 中运行。网络配置策略 NC4 允许 R1 和 R4 在网络资源上享有同等的权利。来自组织 R1 和 R4 的客户端应用程序和对等节点可以通过连接到节点 O1 或节点 O4 来管理网络资源，因为这两个节点的行为方式相同，如网络配置 NC4 中的策略所定义。实际上，来自特定组织的参与者往往使用其上级组织提供的基础结构，但是事实并非总是如此。

去中心化交易分配

排序服务不仅是网络的管理点，而且还提供了另一个关键功能—它是交易的分发点。排序服务是一个组件，它从应用程序中收集已认可的交易并将其排序到交易块中，然后将其分发到通道中的每个对等节点。在这些提交对等方的每个对等方，记录交易（有效或无效），并适当更新其本地分类帐副本。

请注意，排序服务节点 O4 在通道 C1 上的作用与在网络 N 上的作用是非常不同的。当在通道级别进行操作时，O4 的作用是在通道 C1 内收集事务并分配块。它根据通道配置 CC1 中定义的策略执行此操作。相反，当在网络级别执行操作时，O4 的作用是根据网络配置 NC4 中定义的策略为网络资源提供管理点。再次注意，这些角色是如何分别由通道和网络配置中的不同策略定义的。这应该向您增强 Hyperledger Fabric 中基于声明策略的配置的重要性。策略既定义了联盟的每个成员，又用于控制联盟的每个成员的行为。

我们可以看到，排序服务与 Hyperledger Fabric 中的其他组件一样，是完全去中心化的组件。无论是充当网络管理点，还是充当通道中的块分配器，都可以根据需要在网络中的多个组织中分布其节点。

改变政策

在整个示例网络的探索过程中，我们已经了解了控制策略参与者行为的策略的重要性。我们仅讨论了一些可用的策略，但是可以声明性地定义许多策略来控制行为的各个方面。这些单独的策略在文档的其他地方进行了讨论。最重要的是，Hyperledger Fabric 提供了独特而强大的策略，允许网络和渠道管理员自行管理策略更改！基本的哲学思想是，无论是在组织内部还是组织之间发生的变化，还是由外部监管机构施加的变化，变化都是一个常数。例如，新组织可以加入频道，或者现有组织的权限可以增加或减少。让我们进一步研究在 Hyperledger Fabric 中如何实施更改策略。

他们的主要理解点是，策略更改由策略本身内的策略管理和**修改策略**，或 **mod_policy** 的简称，是管理变化的网络或信道配置中的第一类的策略。让我们给的我们如何将两个简短的例子：**已经使用 mod_policy 我们的管理网络中的变化！**

第一个示例是最初建立网络时的情况。此时，仅组织 R4 被允许管理网络。实际上，这是通过使 R4 成为网络配置 NC4 中定义的唯一拥有网络资源许可的组织来实现的。此外，NC4 的 mod_policy 仅提及组织 R4 –仅允许 R4 更改此配置。

然后，我们对网络 N 进行了演进，以允许组织 R1 来管理网络。R4 通过将 R1 添加到用于通道创建和联盟创建的策略中来做到这一点。由于此更改，R1 能够定义联盟 X1 和 X2，并创建通道 C1 和 C2。R1 对网络配置中的通道和联盟策略具有同等的管理权限。

但是，R4 可以通过网络配置为 R1 提供更多的权限！R4 可以将 R1 添加到 mod_policy 中，以便 R1 也能够管理网络策略的更改。

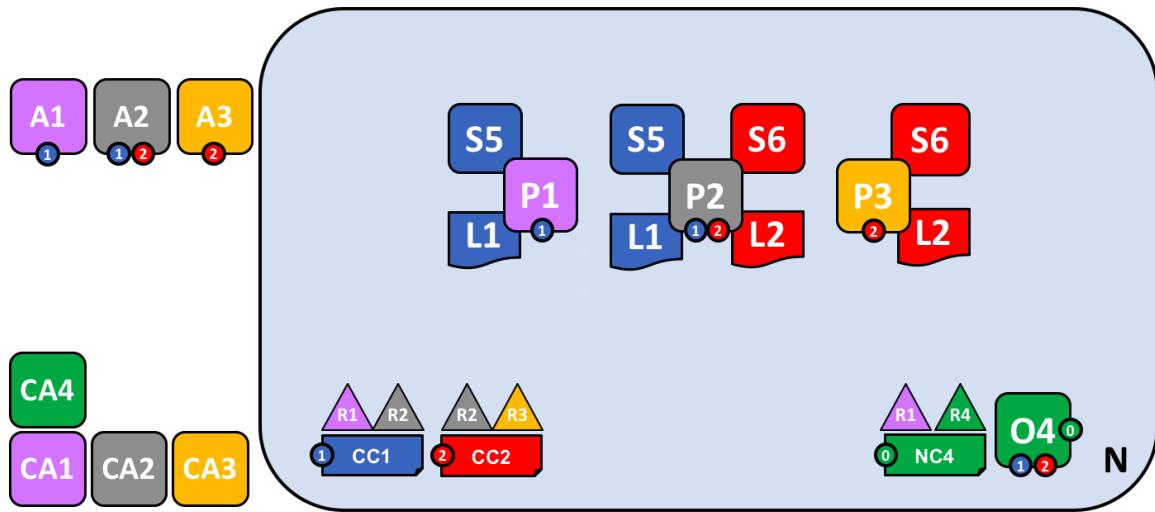
第二个权限比第一个权限强大得多，因为 R1 现在可以 **完全控制** 网络配置 NC4！这意味着 R1 原则上可以从网络中删除 R4 的管理权限。实际上，R4 将配置 mod_policy，以便 R4 也需要批准更改，或者 mod_policy 中的所有组织都必须批准更改。有足够的灵活性可以使 mod_policy 复杂到可以支持所需的任何更改过程。

这是在工作的 mod_policy，它使基本配置可以优雅地演变为复杂的配置。在所有相关组织的同意下，这种情况一直存在。mod_policy 的行为类似于网络或通道配置中的其他所有策略；它定义了一组允许更改 mod_policy 本身的组织。

在本小节中，我们仅涉及了策略和 mod_policy 的功能。在策略主题中将对它进行更详细的讨论，但现在让我们回到完成的网络中！

网络全面形成

让我们使用一致的视觉词汇来回顾一下我们的网络。我们已经使用更紧凑的视觉语法对其进行了稍微的重组，因为它可以更好地适应更大的拓扑：



在此图中，我们看到 Fabric 区块链网络由两个应用程序通道和一个订购通道组成。组织 R1 和 R4 负责订购通道，组织 R1 和 R2 负责蓝色应用程序通道，而组织 R2 和 R3 负责红色应用程序通道。客户端应用程序 A1 是组织 R1 的元素，而 CA1 是组织的证书颁发机构。请注意，组织 R2 的对等方 P2 可以使用蓝色和红色应用程序通道的通信设施。每个应用程序通道都有自己的通道配置，在这种情况下为 CC1 和 CC2。系统通道的通道配置是网络配置 NC4 的一部分。

我们在构建示例 Hyperledger Fabric 区块链网络的概念之旅结束时。我们创建了一个具有两个渠道和三个对等节点，四个智能合约和一个排序服务的四个组织网络。它由四个证书颁发机构支持。它为三个客户端应用程序提供分类帐和智能合约服务，他们可以通过两个渠道与之交互。花一点时间浏览图中的网络详细信息，然后随时阅读本主题以增强您的知识，或者转到更详细的主题。

网络组件摘要

这是我们讨论过的网络组件的简要概述：

- 分类帐。每个频道一个。由 区块链和世界状态组成
- 智能合约（又名链码）
- 对等节点
- 排序服务
- 渠道
- 证书颁发机构

网络汇总

在本主题中，我们已经看到了不同的组织如何共享其基础架构以提供集成的 Hyperledger Fabric 区块链网络。我们已经看到了如何将集体基础结构组织为提供独立管理的私有通信机制的渠道。我们已经看到如何通过使用来自各自证书颁发机构的证书来将诸如客户端应用程序，管理员，对等方和排序节点之类的参与者识别为来自不同组织。反过来，我们已经看到了定义这些组织参与者在网络和渠道资源上拥有的同意权限的策略的重要性。

身份识别

1. 什么是身份?

区块链网络中的不同参与者包括对等方，排序节点，客户端应用程序，管理员等。这些参与者（网络内部或外部能够使用服务的活动元素）中的每一个都有封装在 X.509 数字证书中的数字身份。这些身份确实很重要，因为它们确定了对资源的确切权限以及对参与者在区块链网络中拥有的信息的访问权限。

此外，数字身份还具有 Fabric 用来确定许可权的一些其他属性，并且为身份和关联的属性的并集提供了特殊的名称- **主体**。主体就像 userID 或 groupID 一样，但是更灵活一些，因为它们可以包含参与者身份的各种属性，例如参与者的组织，组织单位，角色，甚至参与者的特定身份。当我们谈论主体时，它们是确定其权限的属性。

要使身份可验证，它必须来自受信任的权威。一[成员的服务提供商](#)（MSP）是在信任的机构。更具体地说，MSP 是定义用于管理该组织的有效身份的规则的组件。Fabric 中的默认 MSP 实现使用 X.509 证书作为身份，并采用传统的公共密钥基础结构（PKI）层次模型（稍后将在 PKI 上进行介绍）。

2. 说明身份使用的简单方案

想象一下，您去一家超市买了一些杂货。在结帐时，您会看到一个标牌，上面只接受 Visa，Mastercard 和 AMEX 卡。如果您尝试使用其他卡付款（我们称其为“ImagineCard”），则该卡是否真实以及帐户中是否有足够的资金都无关紧要。它将不被接受。



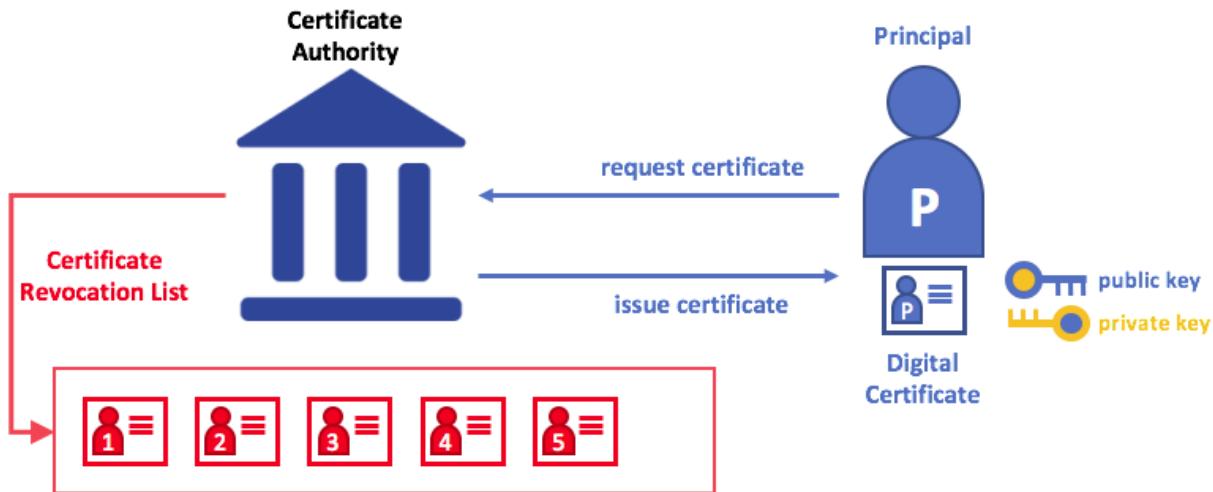
拥有有效的信用卡是不够的-商店也必须接受它！PKI 和 MSP 以相同的方式一起工作-PKI 提供了一个身份列表，而 MSP 表示其中的哪些是参与网络的给定组织的成员。

PKI 证书颁发机构和 MSP 提供了类似的功能组合。PKI 就像卡提供商一样，它分配许多不同类型的可验证身份。另一方面，MSP 就像商店接受的卡提供商列表一样，确定哪些身份是商店支付网络的受信任成员（参与者）。**MSP 将可验证的身份转换为区块链网络的成员。**

让我们更详细地研究这些概念。

3. 什么是 PKI?

公钥基础结构 (PKI) 是在网络中提供安全通信的 Internet 技术的集合。是 PKI 将 S 放入 HTTPS 中 - 如果您正在 Web 浏览器上阅读此文档，则可能正在使用 PKI 来确保它来自经过验证的来源。



公钥基础结构 (PKI) 的元素。PKI 由证书颁发机构组成，证书颁发机构向各方（例如，服务的用户，服务提供商）颁发数字证书，然后由他们使用它们在环境中交换的消息中对自己进行身份验证。CA 的证书吊销列表 (CRL) 构成了不再有效的证书的参考。吊销证书的原因有很多。例如，由于与证书关联的加密专用材料已经暴露，因此证书可能被吊销。

尽管区块链网络不仅仅是通信网络，但它依赖于 PKI 标准来确保各种网络参与者之间的安全通信，并确保正确认证在区块链上发布的消息。因此，了解 PKI 的基本知识以及 MSP 为何如此重要非常重要。

PKI 有四个关键要素：

- 数字证书
- 公钥和私钥
- 证书颁发机构
- 证书吊销列表

让我们快速描述这些 PKI 基础知识，如果您想了解更多详细信息，那么 [Wikipedia](#) 是一个不错的起点。

4. 数字证书

数字证书是一种文档，其中包含与证书持有者有关的一组属性。最常见的证书类型是符合 [X.509 标准](#) 的证书，该证书允许在其结构中对参与方的标识详细信息进行编码。

例如，玛丽·莫里斯米切尔汽车在底特律的制造部，密歇根州可能有一个数字证书具有 **SUBJECT** 的属性 **C=US**，**ST=Michigan**，**L=Detroit**，...。玛丽的证书类似于她的政府身份证件-它提供有关玛丽的信息，她可以用来证明有关她的关键事实。X.509 证书中还有许多其他属性，但现在让我们仅关注这些属性。

O=Mitchell Cars **OU=Manufacturing** **CN=Mary Morris** / **UID=123456**

Mary Morris



```
Certificate:
Data:
Version: 3 (0x2)
Serial Number:
    76:0f:4b:cf:71:2b:a6:95:25:ff:40:aa:67:17:79:0d
Signature Algorithm: ecdsa-with-SHA256
Issuer: C=US, ST=California, L=San Francisco, O=org1.example.com, CN=ca.org1.example.com
Validity
    Not Before: Aug 15 12:24:42 2017 GMT
    Not After : Aug 13 12:24:42 2027 GMT
Subject: C=US, ST=Michigan, L=Detroit, O=Mitchells Cars, OU=Manufacturing, CN=Mary Morris/UID=123456
Subject Public Key Info:
    Public Key Algorithm: id-ecPublicKey
    EC Public Key:
        pub:
            04:5c:0d:b8:d9:f2:e8:9e:d3:aa:85:fe:al:69:44:
            f6:el:6a:bf:dd:3c:3f:6:f8:c5:72:55:01:a2:ca:
            6c:64:b2:da:41:e2:a3:37:2b:d4:a3:9e:bd:41:13:
            ASN1 OID: prime256v1
X509v3 extensions:
    X509v3 Key Usage: critical
        Digital Signature, Key Encipherment, Certificate Sign, CRL Sign
    X509v3 Extended Key Usage:
        2.5.29.37.0
    X509v3 Basic Constraints: critical
        CA:TRUE
    X509v3 Subject Key Identifier:
        51:80:C8:26:FD:02:6A:E4:43:7C:FF:76:56:EA:8F:8C:B0:99:90:F5:F8:AB:6E:1F:
Signature Algorithm: ecdsa-with-SHA256
30:44:02:20:1f:a8:dd:21:b7:33:cc:19:b4:63:cc:aa:a0:ec:
```

描述一个叫做玛丽·莫里斯的政党的数字证书。玛丽是 **SUBJECT** 证书的作者，突出显示的 **SUBJECT** 文本显示了有关玛丽的关键事实。如您所见，该证书还包含更多信息。最重要的是，玛丽的公共密钥分布在她的证书中，而她的私人签名密钥则不是。该签名密钥必须保密。

重要的是，可以使用一种称为密码学的数学技术（字面意义上的“秘密写作”）来记录 Mary 的所有属性，以免篡改证书。只要对方信任证书颁发者（称为**证书颁发机构 (CA)**），密码学就可以允许 Mary 向他人出示她的证书以证明自己的身份。只要 CA 安全地保留某些密码信息（即其自己的**专用签名密钥**），任何阅读证书的人都可以确保有关 Mary 的信息未被篡改-它始终具有 Mary Morris 的那些特定属性。将 Mary 的 X.509 证书视为无法更改的数字身份证件。

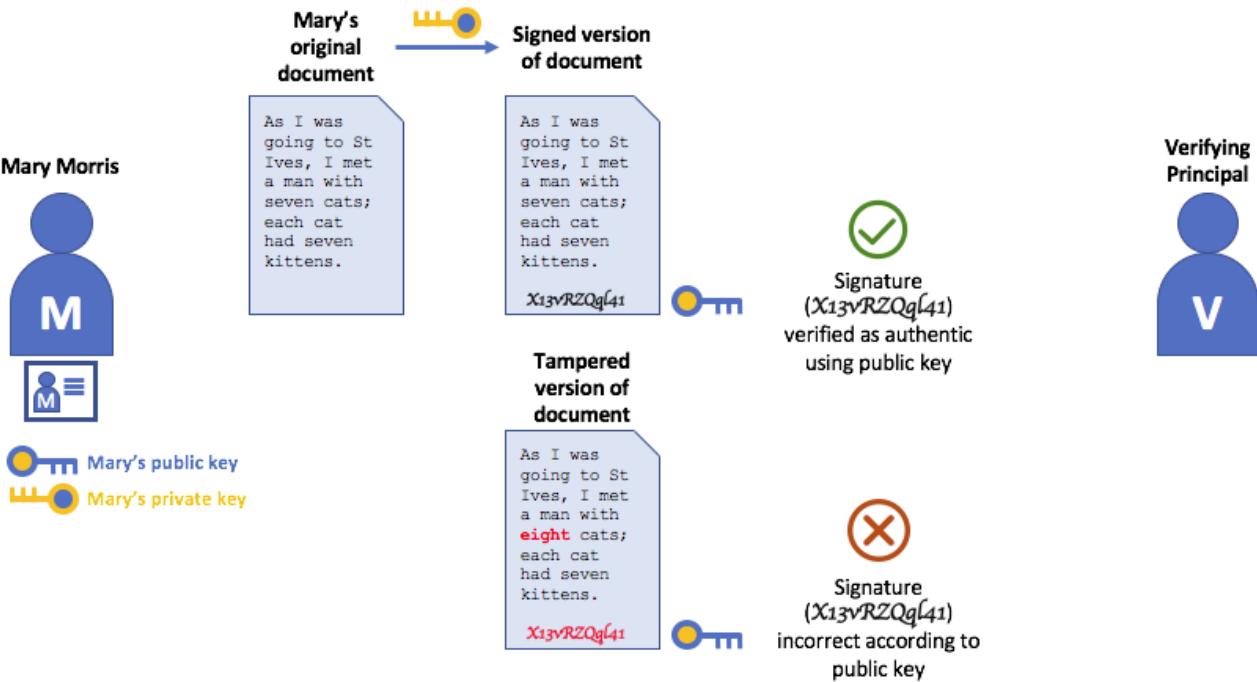
5. 身份验证，公钥和私钥

身份验证和消息完整性是安全通信中的重要概念。身份验证要求交换消息的各方确保创建了特定消息的身份。消息具有“完整性”意味着不能在其传输过程中对其进行修改。例如，您可能要确保与真实的 Mary Morris（而不是模仿者）进行交流。或者，如果 Mary 向您发送了一条消息，则可能要确保在传输过程中没有被其他任何人篡改过该消息。

顾名思义，传统的身份验证机制依赖于**数字签名**，该**数字签名**允许一方对其消息进行**数字签名**。数字签名还为签名消息的完整性提供保证。

从技术上讲，数字签名机制要求每一方都拥有两个加密连接的密钥：一个广泛可用的公共密钥，充当身份验证锚，以及一个用于在消息上产生**数字签名**的私有密钥。经过数字签名的邮件的收件人可以通过检查附加的签名在预期发件人的公钥下是否有效来验证接收到的邮件的来源和完整性。

私钥和相应的公钥之间的独特关系是使安全通信成为可能的加密魔术。密钥之间的独特数学关系使得私钥可用于在仅相应公钥可以匹配的消息上且仅在同一消息上产生签名。

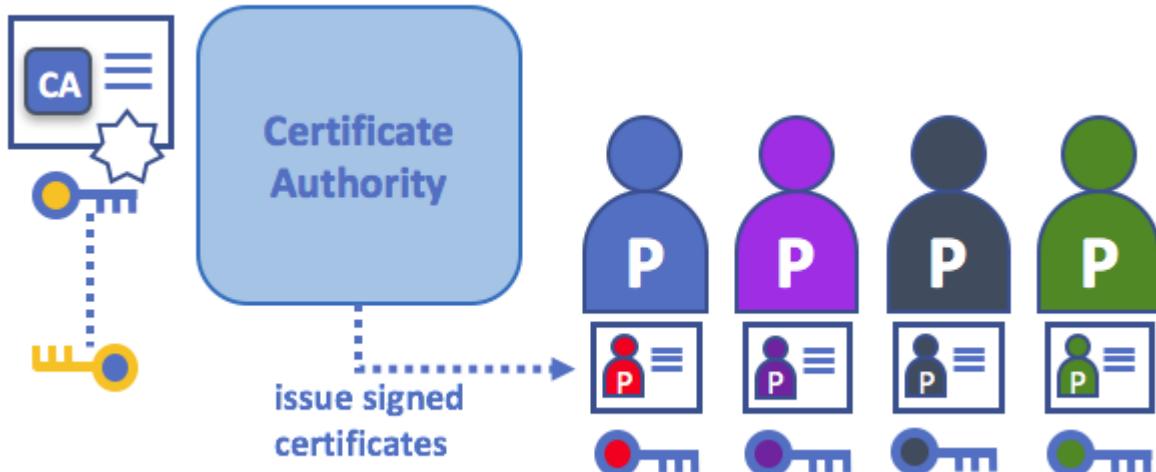


在上面的示例中，Mary 使用她的私钥对邮件签名。可以使用她的公共密钥查看签名消息的任何人来验证签名。

6. 证书颁发机构

如您所见，参与者或节点能够通过系统信任的权限通过为其颁发的**数字身份**来参与区块链网络。在最常见的情况下，**数字身份**（或**简单身份**）具有符合 X.509 标准并由证书颁发机构（CA）颁发的经过密码验证的数字证书的形式。

CA 是 Internet 安全协议的常见部分，您可能已经听说过一些比较流行的协议：Symantec(最初为 Verisign)，GeoTrust，DigiCert，GoDaddy 和 Comodo。



证书颁发机构将证书分发给不同的参与者。这些证书由 CA 进行数字签名，并将角色与角色的公钥（以及可选的完整属性列表）绑定在一起。结果，如果一个人信任 CA（并知道其公钥），则可以通过验证参与者的证书上的 CA 签名，来信任特定角色与证书中包含的公钥绑定，并拥有包含的属性。

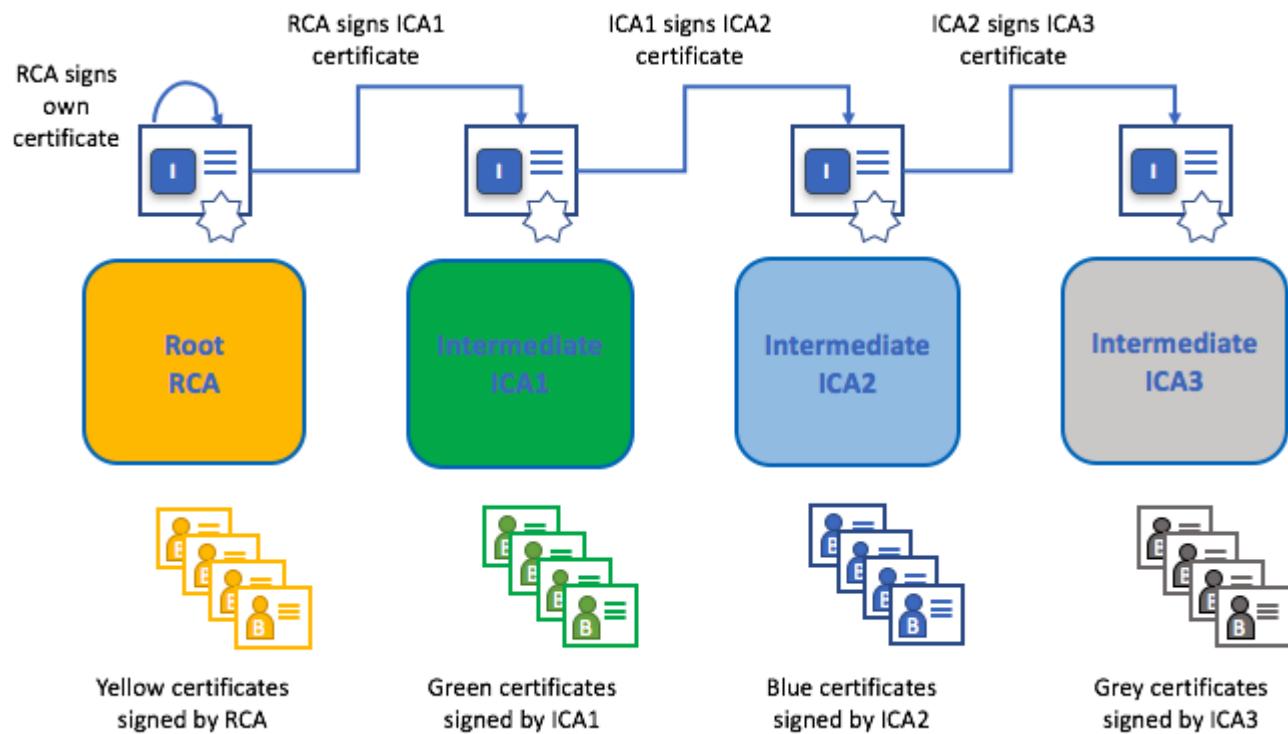
证书可以广泛传播，因为它们既不包含参与者的密钥，也不包含 CA 的私钥。这样，它们可以用作信任的锚，用于验证来自不同参与者的消息。

CA 也有一个证书，可以广泛使用。这使给定 CA 颁发的身份的使用者可以通过检查证书仅由相应私钥（CA）的持有者生成来验证它们。

在区块链环境中，每个希望与网络交互的参与者都需要一个身份。在这种情况下，您可能会说一个或多个 CA 可以用于从数字角度定义组织的成员。正是 CA 为组织的参与者提供了可验证的数字身份提供了基础。

7. 根 CA，中间 CA 和信任链

CA 的有两种形式：**根 CA** 和**中间 CA**。由于根 CA（赛门铁克，Geotrust 等）必须向互联网用户安全地分发数亿个证书，因此有必要在所谓的**中间 CA**中分散该过程。这些中间 CA 的证书由根 CA 或其他中间机构颁发，从而可以为链中任何 CA 颁发的任何证书建立“信任链”。这种追溯到根 CA 的功能不仅可以扩展 CA 的功能，同时仍提供安全性-允许使用证书的组织放心使用中间 CA，它还可以限制根 CA 的暴露范围，如果受到损害，则会危害整个信任链。另一方面，如果中级 CA 受到威胁，则风险会小得多。



只要用于这些中间 CA 的证书的颁发 CA 是根 CA 本身或对根 CA 有信任链，就可以在根 CA 与一组中间 CA 之间建立信任链。

跨多个组织颁发证书时，中间 CA 提供了极大的灵活性，这在许可的区块链系统（如 Fabric）中非常有帮助。例如，您将看到不同的组织可能使用不同的根 CA，或者同一根 CA 使用不同的中间 CA，这确实取决于网络的需求。

8. Fabric CA

因为 CA 非常重要，所以 Fabric 提供了内置的 CA 组件，允许您在形成的区块链网络中创建 CA。该组件（称为**Fabric CA**）是私有的根 CA 提供者，能够管理具有 X.509 证书形式的 Fabric 参与者的数字身份。由于 Fabric CA 是针对 Fabric 的根 CA 需求的自定义 CA，因此它固有地无法提供 SSL 证书供浏览器中的常规/自动使用。但是，由于必须

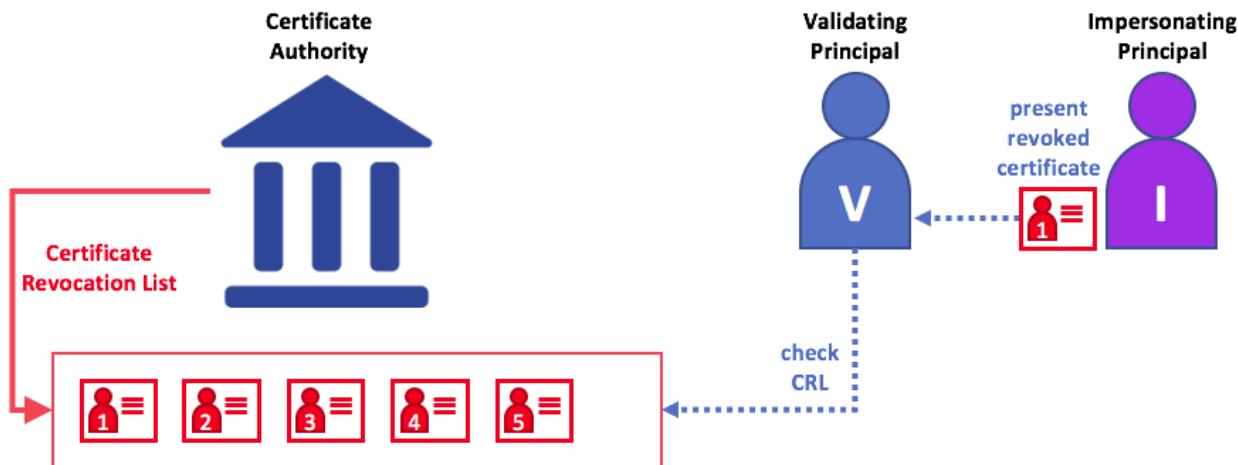
使用某些 CA 来管理身份（即使在测试环境中也是如此），因此可以将 Fabric CA 用于提供和管理证书。使用公共/商业根或中间 CA 进行标识也是可能的，并且是完全合适的。

如果您有兴趣，可以在“[CA 文档](#)”部分中阅读有关 Fabric CA 的更多信息。

9. 证书吊销列表

证书吊销列表 (CRL) 易于理解-只是 CA 已知由于某种原因而吊销的证书引用列表。如果您回想起商店的情况，CRL 就像一张被盗信用卡的清单。

当第三方想要验证另一方的身份时，它首先检查签发的 CA 的 CRL，以确保证书未被吊销。验证者不必检查 CRL，但如果这样做，则冒接受被泄露身份的风险。



使用 CRL 检查证书仍然有效。如果冒名顶替者试图将已泄露的数字证书传递给验证方，则可以首先与颁发 CA 的 CRL 进行核对，以确保它不再列为不再有效。

请注意，被吊销的证书与证书到期有很大不同。吊销的证书尚未过期-通过其他方法，它们是完全有效的证书。有关 CRL 的更多详细信息，请单击[此处](#)。

既然您已经了解了 PKI 如何通过信任链提供可验证的身份，下一步就是了解如何使用这些身份来表示区块链网络的受信任成员。这就是会员服务提供商 (MSP) 发挥作用的地方- **它标识了区块链网络中给定组织的成员的参与方。**

要了解有关成员资格的更多信息，请查看有关 [MSP](#) 的概念性文档。

会员资格

如果您已阅读有关身份的文档，那么您将了解 PKI 如何通过信任链提供可验证的身份。现在让我们看看如何使用这些身份来表示区块链网络的受信任成员。

这是**成员资格服务提供商 (MSP)** 发挥作用的地方- 它通过列出成员的身份或通过标识来确定哪些根 CA 和中间 CA 受信任来定义信任域 (例如组织) 的成员。哪些 CA 被授权为其成员发布有效身份，或者 (通常是这样) 通过将两者结合使用。

MSP 的功能不仅限于列出谁是网络参与者或渠道成员。MSP 可以识别参与者可能在 MSP 代表的组织范围内 (例如，管理员或作为子组织组的成员) 扮演的特定角色，并为在网络上下文中定义访问权限设置基础和渠道 (例如，渠道管理员，读者，作家)。

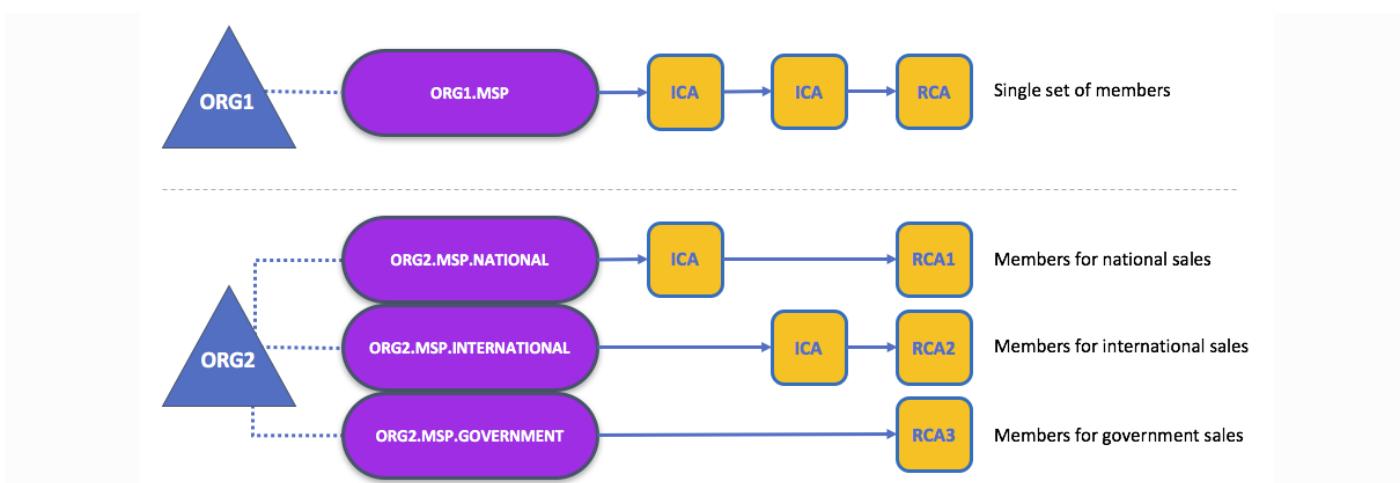
将 MSP 的配置发布到相应组织的成员参与的所有渠道 (以渠道 MSP 的形式)。除了通道 MSP 外，对等节点，排序者和客户端还维护**本地 MSP**，以在通道上下文之外对成员消息进行身份验证，并定义对特定组件的权限 (该组件可以在对等方上安装链码，用于例)。

另外，如身份 文档中所讨论的，MSP 可以识别已撤销的身份列表，但是我们将讨论该过程如何扩展到 MSP。我们稍后将详细讨论本地和渠道 MSP。现在，让我们看看 MSP 的一般功能。

1. 将 MSP 映射到组织

一个**组织**的成员的管理组。它可以像跨国公司一样大，也可以像花店一样小。典型地，组织被表示为单个 MSP。请注意，这与 X.509 证书中定义的组织概念不同，我们稍后将讨论。

组织与其 MSP 之间的排他性关系使得以该组织命名 MSP 是很明智的，您会在大多数策略配置中都采用该约定。例如，组织 **ORG1** 可能会有一个 MSP 之类的 MSP **ORG1-MSP**。在某些情况下，组织可能需要多个成员资格组-例如，在组织之间使用渠道执行非常不同的业务功能的情况下。在这种情况下是有意义的有多个 MSP 和因此他们的名字，例如，**ORG2-MSP-NATIONAL** 和 **ORG2-MSP-GOVERNMENT**，反映了内部的信任的不同成员的根 **ORG2** 在 **NATIONAL** 相对于销售渠道 **GOVERNMENT** 的监管渠道。



组织的两种不同的 **MSP** 配置。第一个配置显示了 **MSP** 与组织之间的典型关系-单个 **MSP** 定义组织的成员列表。在第二种配置中，使用不同的 **MSP** 代表具有国家，国际和政府从属关系的不同组织组。

2. 组织单位和 MSP

一个组织通常分为多个**组织单位** (OU)，每个**组织单位**都有一定的职责集。例如，**ORG1** 组织可能同时具有 **ORG1-MANUFACTURING** 和 **ORG1-DISTRIBUTION** OU 来反映这些单独的业务线。当 CA 颁发 X.509 证书时，证书中的 **OU** 字段指定身份所属的业务范围。

稍后我们将看到 **OU** 如何帮助控制被视为区块链网络成员的组织部分。例如，只有来自 **ORG1-MANUFACTURING** OU 的身份才可以访问通道，而 **ORG1-DISTRIBUTION** 不能。

最后，尽管这是对 **OU** 的轻微滥用，但是有时联合体中的不同组织可以使用它们来区分彼此。在这种情况下，不同的组织将相同的根 CA 和中间 CA 用于其信任链，但会分配该 **OU** 字段以标识每个组织的成员。我们还将在以后看到如何配置 **MSP** 来实现此目的。

3. 本地和渠道 MSP

MSP 出现在区块链网络中的两种类型的位置中：

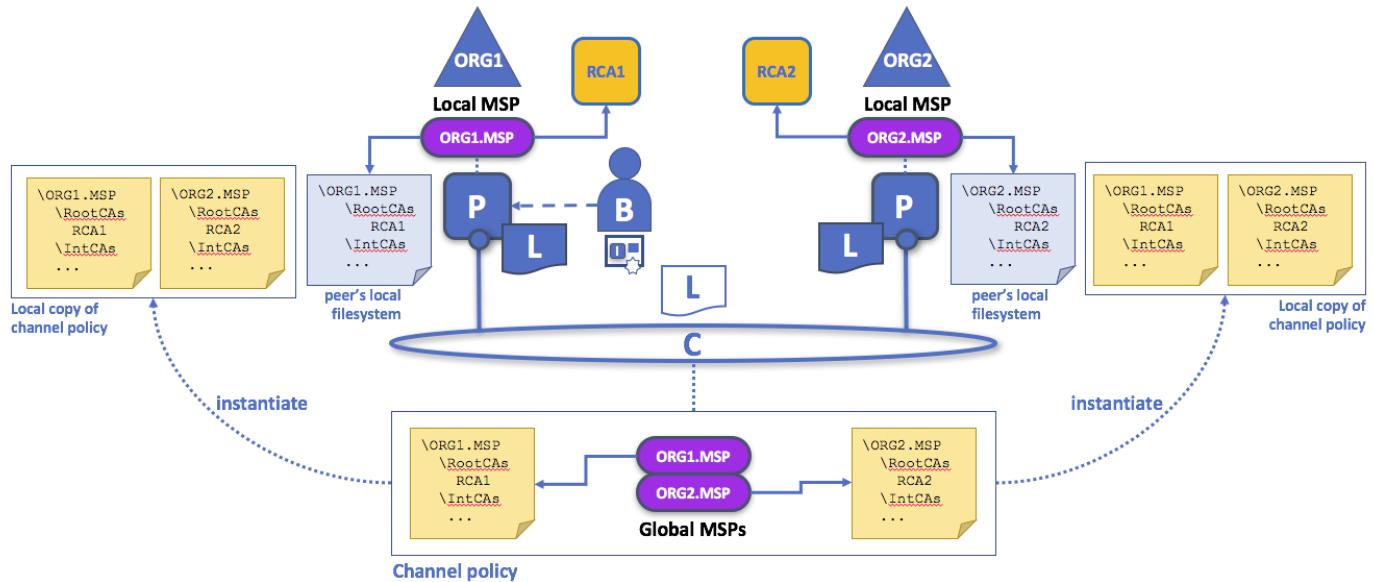
- 在通道配置中 (**通道 MSP**)
- 本地执行者的节点上 (**本地 MSP**)

为客户端（用户）和节点（对等和排序者）定义了本地 MSP。本地 MSP 定义了该节点的权限（例如，对等管理员是谁）。用户的本地 MSP 允许用户端在其交易中作为渠道成员（例如，在链码交易中）或作为系统中特定角色的所有者（例如，组织管理员）进行身份验证交易）。

每个节点和用户都必须定义一个本地 MSP，因为它定义了在该级别具有管理或参与权的人（同级管理员不一定是渠道管理员，反之亦然）。

相反，**渠道 MSP 在渠道级别定义了管理权和参与权**。每个参与渠道的组织都必须为其定义 **MSP**。通道上的对等方和排序节点将共享相同的通道 **MSP** 视图，因此将能够正确地验证通道参与者。这意味着，如果组织希望加入渠道，则需要在渠道配置中包含一个包含组织成员信任链的 **MSP**。否则，来自该组织身份的交易将被拒绝。

本地 **MSP** 和渠道 **MSP** 之间的主要区别不是它们的功能（都将身份转变为角色），而是它们的**范围**。



本地和渠道 *MSP*。每个对等方的信任域（例如，组织）由对等方的本地 *MSP*（例如，*ORG1* 或 *ORG2*）定义。通过在通道配置中添加组织的 *MSP*，可以实现组织在通道上的表示。例如，此图的通道同时由 *ORG1* 和 *ORG2* 管理。类似的原理适用于网络，排序节点和用户，但为简单起见，此处未显示。

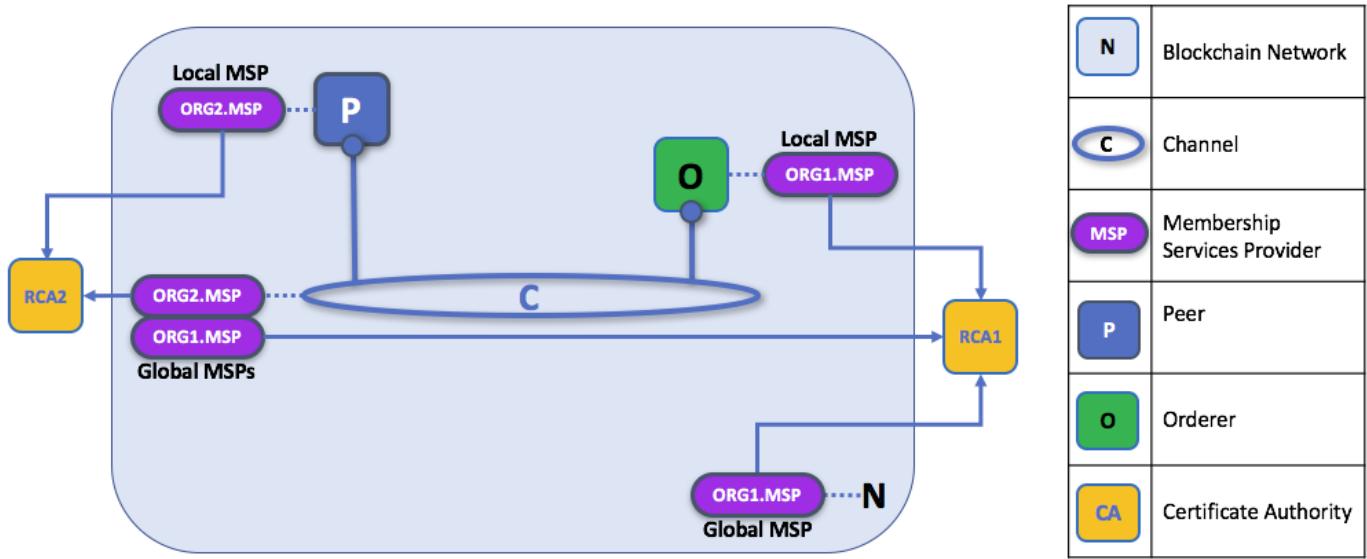
您可能会发现通过查看区块链管理员安装和实例化智能合约时会发生什么，了解使用本地和通道 *MSP* 很有帮助，如上图所示。

管理员 **B** 使用 **RCA1** 其本地 *MSP* 发出并存储在其本地 *MSP* 中的身份连接到对等方。当 **B** 尝试在对等方上安装智能合约时，对等方会检查其本地 *MSP* **ORG1-MSP**，以验证的身份 **B** 确实是的成员 **ORG1**。成功的验证将使 *install* 命令成功完成。随后，**B** 希望在通道上实例化智能合约。因为这是渠道操作，所以渠道上的所有组织都必须对此表示同意。因此，对等方必须先检查通道的 *MSP*，然后才能成功提交此命令。（其他事情也必须发生，但现在就集中在上面。）

本地 *MSP* 仅在它们应用到的节点或用户的文件系统上定义。因此，在物理上和逻辑上，每个节点或用户只有一个本地 *MSP*。但是，由于通道 *MSP* 可用于通道中的所有节点，因此它们在通道配置中逻辑定义一次。但是，**通道 *MSP* 也会在通道中每个节点的文件系统上实例化，并通过共识保持同步。**因此，尽管在每个节点的本地文件系统上都有每个通道 *MSP* 的副本，但从逻辑上讲，通道 *MSP* 驻留在通道或网络上并由通道或网络维护。

4. MSP 级别

通道和本地 *MSP* 之间的划分反映了组织管理在通道或网络级别运行的本地资源（例如对等节点或排序者节点）以及其通道资源（例如分类帐，智能合约和联盟）的需求。可以将这些 *MSP* 视为不同**级别**，这很有帮助，其中**较高级别的 *MSP* 与网络管理问题有关，而较低级别的 *MSP* 处理身份管理私有资源**。*MSP* 在每个管理级别都是必需的--必须为网络，渠道，对等方，排序者和用户定义 *MSP*。

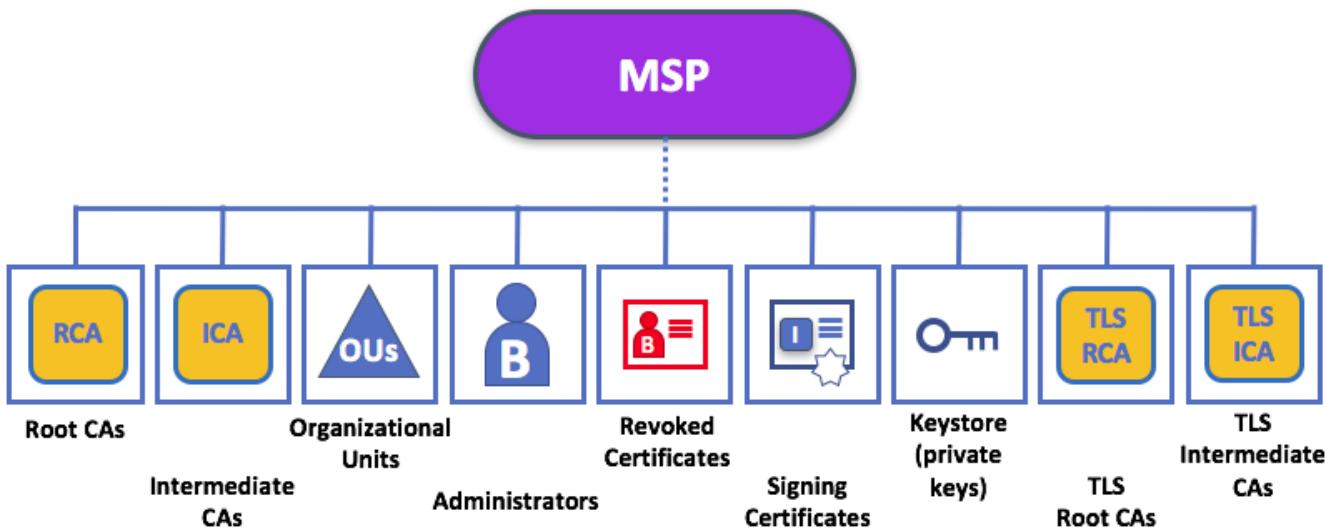


MSP 级别。 对等方和排序者的 **MSP** 是本地的，而某个通道（包括网络配置通道）的 **MSP** 在该通道的所有参与者之间共享。在此图中，网络配置通道由 **ORG1** 管理，但另一个应用程序通道可以由 **ORG1** 和 **ORG2** 管理。对等方是 **ORG2** 的成员并由其管理，而 **ORG1** 管理图的顺序。**ORG1** 信任来自 **RCA1** 的身份，而 **ORG2** 信任来自 **RCA2** 的身份。请注意，这些是管理身份，反映了谁可以管理这些组件。因此，当 **ORG1** 管理网络时，**ORG2.MSP** 确实存在于网络定义中。

- 网络 MSP：** 网络的配置通过定义参与组织的 **MSP** 来定义谁是网络中的成员，以及授权这些成员中的哪些成员执行管理任务（例如，创建频道）。
- 频道 MSP：** 频道要分别维护其成员的 **MSP**，这一点很重要。通道在一组特定的组织之间提供私人通信，而这些组织又对该组织具有管理控制权。在该通道的 **MSP** 上下文中解释的通道策略定义了谁有能力参与该通道上的某些操作，例如，添加组织或实例化链码。请注意，管理通道的权限与管理网络配置通道（或任何其他通道）的能力之间没有必要的关系。行政权利存在于所管理的范围内（除非规则另有规定，请参见 **ROLE** 下面有关属性的讨论）。
- 对等 MSP：** 此本地 **MSP** 在每个对等方的文件系统上定义，并且每个对等方都有一个 **MSP** 实例。从概念上讲，它执行与通道 **MSP** 完全相同的功能，但限制是它仅适用于定义它的对等方。使用对等方的本地 **MSP** 评估授权的操作示例是在对等方上安装链码。
- 排序者 MSP：** 与对等 **MSP** 一样，**排序本地 MSP** 也定义在节点的文件系统上，并且仅适用于该节点。像对等节点一样，排序者也由单个组织拥有，因此具有单个 **MSP** 来列出其信任的参与者或节点。

10. MSP 结构

到目前为止，您已经看到，**MSP** 的最重要元素是根或中间 **CA** 的规范，用于在各个组织中建立参与者或节点的成员资格。但是，还有更多元素与这两个元素结合使用以辅助成员资格功能。



上图显示了本地 MSP 如何存储在本地文件系统上。即使通道 MSP 的物理结构不是完全以这种方式构造的，它仍然是一种思考它们的有用方法。

如您所见，MSP 有九个要素。在目录结构中最容易想到这些元素，其中 MSP 名称是根文件夹名称，每个子文件夹代表 MSP 配置的不同元素。

让我们更详细地描述这些文件夹，看看它们为什么很重要。

- **根 CA:** 此文件夹包含由此 MSP 代表的组织信任的根 CA 的自签名 X.509 证书的列表。此 MSP 文件夹中必须至少有一个 Root CA X.509 证书。

这是最重要的文件夹，因为它标识了必须从其派生所有其他证书才能被视为相应组织的成员的 CA。

- **中间 CA:** 此文件夹包含此组织信任的中间 CA 的 X.509 证书的列表。每个证书必须由 MSP 中的一个根 CA 签名，或由其发行 CA 链最终引回到受信任的根 CA 的中间 CA 签名。

中间 CA 可以代表组织的不同细分（例如 **ORG1-MANUFACTURING** 和 **ORG1-DISTRIBUTION** 为 **ORG1**），也可以代表组织本身（如果商业 CA 被用于组织的身份管理，则可能是这种情况）。在后一种情况下，中间 CA 可以用来表示组织细分。在这里，您可以找到有关 MSP 配置最佳实践的更多信息。请注意，可能有一个没有中间 CA 的正常运行的网络，在这种情况下，此文件夹将为空。

与“根 CA”文件夹类似，此文件夹定义了必须颁发证书才能被视为组织成员的 CA。

- **组织单位 (OU) :** 这些 **\$FABRIC_CFG_PATH/msp/config.yaml** 文件在文件中列出，并包含组织单位的列表，其成员被认为是此 MSP 代表的组织的一部分。当您希望将组织的成员限制为持有具有特定 OU 的身份（由 MSP 指定的 CA 之一签名）的成员时，此功能特别有用。

指定 OU 是可选的。如果未列出任何 OU，则 MSP 一部分的所有身份（由根 CA 和中间 CA 文件夹标识）将被视为组织的成员。

- **管理员:** 此文件夹包含标识列表，这些标识定义了具有该组织管理员角色的参与者。对于标准 MSP 类型，此列表中应该有一个或多个 X.509 证书。

值得注意的是，仅仅因为执行者具有管理员角色，并不意味着他们可以管理特定资源！给定身份在管理系统方面的实际能力由管理系统资源的策略确定。例如，渠道策略可能指定 **ORG1-MANUFACTURING** 管理员有权将新组织添加到渠道，而 **ORG1-DISTRIBUTION** 管理员则没有这种权利。

即使 X.509 证书具有 **ROLE** 属性（例如，指定参与者是的 **admin**），这仍指参与者在其组织内而不是在区块链网络上的角色。这类似于 **OU** 属性的目的，该属性（如果已定义）是指参与者在组织中的位置。

如果已编写该通道的策略以允许组织（或某些组织）中的任何管理员执行某些通道功能（例如实例化链码）的权限，则该 **ROLE** 属性可用于在该通道级别授予管理权限。这样，组织角色可以赋予网络角色。

- **吊销证书：**如果演员的身份已被吊销，则有关身份的标识信息（而不是身份本身）保存在此文件夹中。对于基于 X.509 的身份，这些标识符是称为主题密钥标识符（SKI）和授权访问标识符（AKI）的字符串对，并且只要使用 X.509 证书来确保未使用该证书，就对其进行检查。被撤销。

此列表在概念上与 CA 的证书吊销列表（CRL）相同，但它也与组织中的成员资格吊销有关。结果，MSP 的管理员（本地或渠道）可以通过向 CA 的更新的 CRL 发布由其颁发的吊销证书来快速吊销组织中的参与者或节点。此“列表列表”是可选的。仅当证书被吊销时，它才会被填充。

- **节点身份：**此文件夹包含节点的身份，即，与内容结合使用的加密材料，**KeyStore** 将允许节点在发送给其通道和网络其他参与者的消息中对自身进行身份验证。对于基于 X.509 的身份，此文件夹包含 **X.509 证书**。例如，这是对等方在交易提议响应中放置的证书，以指示对等方已认可该证书-随后可以在验证时根据结果交易的背书策略检查该证书。

对于本地 MSP，此文件夹是必需的，并且该节点必须完全有一个 X.509 证书。它不用于通道 MSP。

- **KeyStore 用于私钥：**此文件夹是为对等节点或排序节点节点的本地 MSP（或在客户端的本地 MSP 中）定义的，并且包含节点的**签名密钥**。该密钥在密码上匹配包含在“**节点标识**”文件夹中的**节点的标识**，并用于对数据进行签名-例如，作为签署阶段的一部分，对交易建议响应进行签名。

此文件夹对于本地 MSP 是必需的，并且必须仅包含一个私钥。显然，对此文件夹的访问必须仅限于对对等具有管理责任的用户的身份。

通道 MSP 的配置不包括此文件夹，因为通道 MSP 仅旨在提供身份验证功能而不是签名功能。

- **TLS 根 CA：**此文件夹包含此组织信任**用于 TLS 通信**的根 CA 的自签名 X.509 证书的列表。TLS 通信的一个示例是当对等方需要连接到排序节点以便接收分类账更新时。

MSP TLS 信息与网络内部的节点有关-换句话说，对等方和排序节点，而不是消耗网络的应用程序和管理。此文件夹中至少必须有一个 TLS 根 CA X.509 证书。

- **TLS 中间 CA：**此文件夹包含此 MSP 代表的组织**用于 TLS 通信**的受信任中间 CA 证书的列表。当商业 CA 用于组织的 TLS 证书时，此文件夹特别有用。与成员资格中间 CA 相似，指定中间 TLS CA 是可选的。

有关 TLS 的更多信息，请单击[此处](#)。

如果您已经阅读了本文档以及有关 **Identity** 的文档，则应该对 Hyperledger Fabric 中的身份和成员身份工作有很好的了解。您已经了解了如何使用 PKI 和 MSP 来识别在区块链网络中进行协作的参与者。您已经了解了证书，公钥/私钥和信任根的工作原理，以及 MSP 的物理和逻辑结构。

政策规定

受众：架构师，应用程序和智能合约开发人员，管理员

在本主题中，我们将介绍：

- [什么是政策](#)
- [为什么需要政策](#)
- [如何在整个 Fabric 中实施政策](#)
- [架构策略域](#)
- [您如何在 Fabric 中编写策略](#)
- [光纤链代码生命周期](#)
- [覆盖策略定义](#)

1. 什么是政策

在最基本的级别上，策略是一组规则，这些规则定义了如何制定决策和达到特定结果的结构。为此，策略通常描述了**谁**和**什么**，例如个人对**资产**的访问或权利。我们可以看到，政策在我们的日常生活中一直被用来保护对我们有价值的资产，使其免受汽车租赁，医疗，房屋等的侵害。

例如，保险单定义了将在其下进行保险赔付的条件，条款，限额和期限。该保单经保单持有人和保险公司同意，并定义了各方的权利和责任。

尽管已为风险管理制定了保险单，但在 Hyperledger Fabric 中，保险单是基础架构管理的机制。架构策略代表成员如何就接受或拒绝对网络，渠道或智能合约的更改达成协议。最初配置网络时，策略成员会获得联盟成员的同意，但是也可以随着网络的发展而修改策略。例如，它们描述了在通道中添加或删除成员，更改块的形成方式或指定认可智能合约所需的组织数量的标准。所有这些动作均由定义了谁可以执行该动作的策略来描述。简而言之，您要在 Fabric 网络上执行的所有操作均受策略控制。

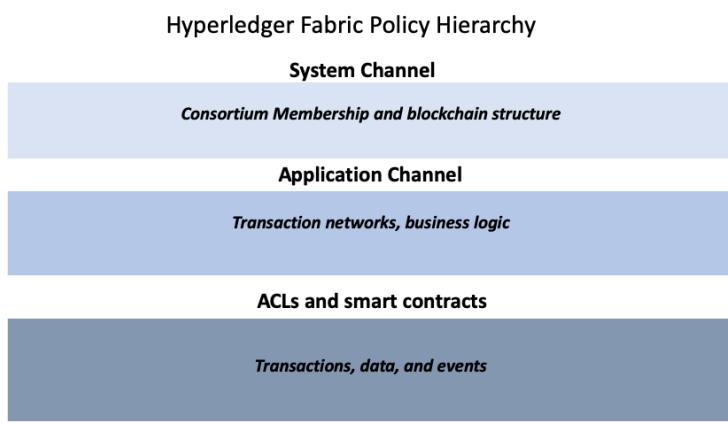
2. 为什么需要政策

政策是使 Hyperledger Fabric 与以太坊或比特币等其他区块链有所不同的原因之一。在那些系统中，交易可以由网络中的任何节点生成和验证。支配网络的策略在任何时间点都是固定的，只能使用支配代码的相同过程进行更改。由于 Fabric 是许可的区块链，其用户被基础基础结构识别，因此这些用户具有在启动网络之前确定网络治理并更改运行中网络的治理的能力。

策略允许成员决定哪些组织可以访问或更新结构网络，并提供执行这些决定的机制。策略包含有权访问给定资源（例如用户或系统链码）的组织的列表。他们还指定了多少组织需要就更新资源（例如渠道或智能合约）的提案达成一致。一旦写入，策略将评估交易和投标所附签名的集合，并验证签名是否满足网络所同意的治理。

3. 如何在整个 Fabric 中实施政策

策略是在 Fabric 网络的不同级别上实施的。每个策略域管理网络运行方式的不同方面。



结构策略层次结构的直观表示。

4. 系统通道配置

每个网络都从排序节点的**系统通道**开始。排序服务必须有一个准确的排序系统通道，这是要创建的第一个通道。系统通道还包含作为排序服务成员的组织（排序组织）和在网络上进行交易的组织（联盟组织）。

排序节点的系统通道配置块中的策略控制排序服务使用的共识，并定义如何创建新块。系统通道还控制允许联盟的哪些成员创建新通道。

5. 应用通道配置

应用程序通道用于在联盟中的组织之间提供专用的通信机制。

应用程序通道中的策略控制从通道中添加或删除成员的能力。应用程序通道还控制使用 Fabric 链码生命周期定义链码并将其提交给通道之前，需要哪些组织批准链码。最初创建应用程序通道时，默认情况下它将继承定购者系统通道中的所有定购服务参数。但是，可以在每个通道中自定义这些参数（以及控制它们的策略）。

6. 访问控制列表 (ACL)

网络管理员将对 ACL 的 Fabric 使用特别感兴趣，ACL 可以通过将资源与现有策略相关联来配置对资源的访问。这些“资源”可以是系统链代码（例如，“qscc”系统链代码上的“GetBlockByNumber”）上的功能，也可以是其他资源（例如可以接收 Block 事件的资源）上的功能。ACL 引用在应用程序通道配置中定义的策略，并将其扩展以控制其他资源。默认的 Fabric ACL 集 `configtx.yaml` 在该部分下的文件中可见，但是可以并且应该在生产环境中覆盖它们。命名为的资源列表是 Fabric 当前定义的所有内部资源的完整集合。

`Application: &ApplicationDefaultsconfigtx.yaml`

在该文件中，ACL 使用以下格式表示：

```
# ACL policy for chaincode to chaincode invocation
peer/ChaincodeToChaincode: /Channel/Application/Readers
```

其中 `peer/ChaincodeToChaincode` 代表受保护的资源，`/Channel/Application/Readers` 是指为使关联交易被视为有效而必须满足的策略。

要深入了解 ACLS，请参阅《[ACL 操作指南](#)》中的主题。

7. 智能合约背书政策

Chaincode 包中的每个智能合约都有一个签注策略，该策略指定了多少个属于不同渠道成员的对等方需要针对给定的智能合约执行和验证交易，以使该交易被视为有效。因此，背书策略定义了必须（通过其对等节点）“批准”（即批准）提案执行的组织。

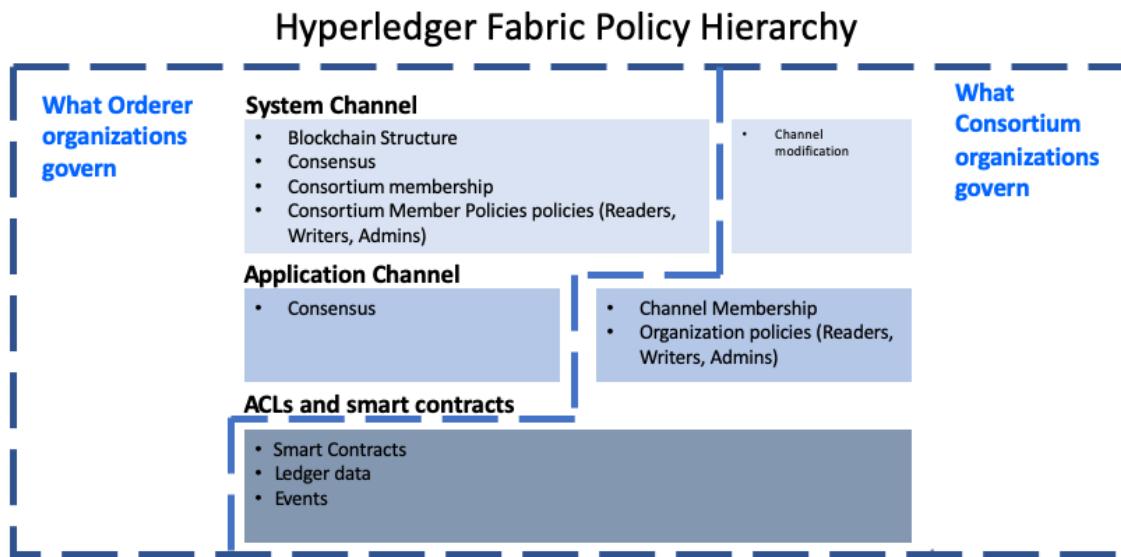
8. 修改政策

最后一种策略对 Fabric 中策略的工作方式至关重要。修改策略指定签署（批准）任何配置更新所需的一组身份。该策略定义了如何更新策略。因此，每个信道配置元素包括对管理其修改的策略的引用。

Modification policy

9. 政策领域

虽然结构策略是灵活的并且可以配置为满足网络的需求，但是策略结构自然会导致由排序服务组织或联盟成员管理的域之间的划分。在下图中，您可以在下面查看默认策略如何实现对结构策略域的控制。



更详细地了解由排序节点组织和联盟组织管理的策略域。

功能齐全的 Fabric 网络可以使许多组织承担不同的责任。域允许排序服务的创建者建立初始规则和联盟成员的能力，从而能够将不同的特权和角色扩展到不同的组织。它们还允许加入该联盟的组织创建私有应用程序通道，管理他们自己的业务逻辑，并限制对网络上放置的数据的访问。

系统通道配置和每个应用程序通道配置的一部分为排序组织提供控制，这些组织可以控制哪些组织是联盟的成员，如何将块交付到渠道以及排序服务的节点所使用的共识机制。

系统通道配置使联盟成员能够创建通道。应用程序通道和 ACL 是联盟组织用来在通道中添加或删除成员并限制对通道上的数据和智能合约的访问的机制。

10. 您如何在 Fabric 中编写策略

如果要更改 Fabric 中的任何内容，则与资源相关联的策略描述了谁需要批准它，可以是个人的显式退出，也可以是组的隐式退出。在保险领域，明确的签名可以是房主保险代理人组的单个成员。隐式签字类似于要求房主保险

集团的大多数管理人员批准。这特别有用，因为该组的成员可以随时间变化而无需更新策略。在 Hyperledger Fabric 中，策略中的显式签发使用 `Signature` 语法表示，隐式签发使用 `ImplicitMeta` 语法表示。

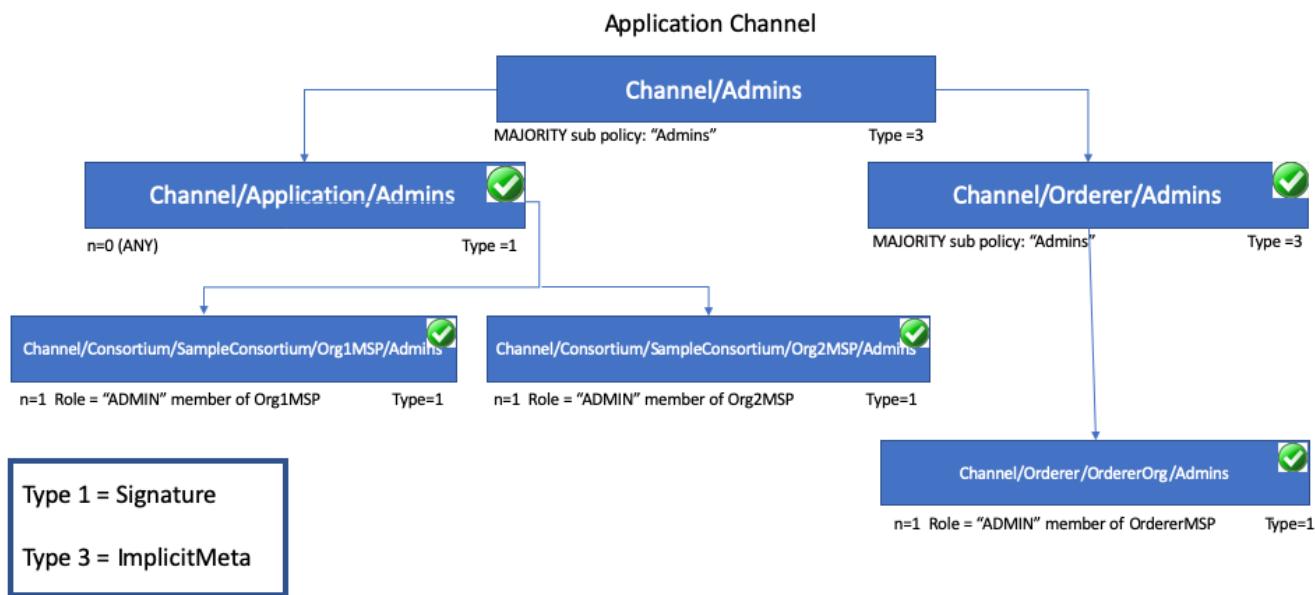
11. 签名政策

`Signature` 策略定义了必须签名才能满足策略的特定用户类型，例如。这些策略被认为是最通用的，因为它们允许构建极其特定的规则，例如：“组织 A 的管理员和其他 2 位管理员，或 6 位组织管理员中的 5 位”。语法支持的任意组合，和。例如，通过使用可以轻松表达策略，这意味着要满足该策略，需要来自 Org1 中至少一个成员和 Org2 中一个成员的签名。`Org1.Peer OR Org2.Peer AND OR NOT OutOf AND (Org1, Org2)`

12. 隐式元策略

`ImplicitMeta` 策略仅在通道配置的上下文中有效，该配置基于配置树中策略的分层层次结构。`ImplicitMeta` 策略在配置树中更深地聚合策略的结果，这些策略最终由签名策略定义。这是 `Implicit` 因为它们是基于通道配置中的当前组织隐式构造的，并且是 `Meta` 因为它们的评估不是针对特定的 MSP 主体，而是针对配置树中其下的其他子策略。

下图说明了应用程序通道的分层策略结构，并显示了满足配置层次结构中位于其下方的子策略（每个复选标记表示子条件）时 `ImplicitMeta`，如何 `/Channel/Admins` 解析名为的通道配置管理员策略。`Admins` 政策很满意。



In order for the `Channel/Admins` policy to be satisfied, every sub-policy under it in the configuration hierarchy must be satisfied.

如上图所示，`ImplicitMeta` Type = 3 的策略使用不同的语法，例如：

```
"<ANY|ALL|MAJORITY> <SubPolicyName>"  
`MAJORITY sub policy: Admins`
```

该图显示了一个子策略 `Admins`，该子策略引用 `Admins` 配置树中其下方的所有策略。您可以创建自己的子策略，并根据需要命名它们，然后在每个组织中对其进行定义。

如上所述，这种 `ImplicitMeta` 策略的主要优点是，当您将新的管理组织添加到通道时，不必更新通道策略。因此，随着联盟成员的变化，政策被认为更加灵活。当添加新成员或现有成员离开时，排序者上的联盟可以更改，联盟成员同意更改，但是不需要更新策略。回想一下，策略最终会在配置树中解决其下方的子策略，

如图所示。 MAJORITY AdminsImplicitMetaImplicitMetaSignature

您还可以定义一个应用程序级别的隐式策略，以在组织中（例如，在一个渠道中）进行操作，并要求满足它们中的任何一个，满足所有条件或满足大多数条件。这种格式适合于更好，更自然的默认设置，以便每个组织可以决定对有效背书的含义。

如果您包含 NodeOUs 在组织定义中，则可以实现进一步的粒度和控制。组织单位（OU）在 Fabric CA 客户端配置文件中定义，并且在创建时可以与身份相关联。在 Fabric 中，NodeOUs 提供一种在数字证书层次结构中对身份进行分类的方法。例如，NodeOUs 启用了特定功能的组织可能要求“对等节点”签名才能获得有效的认可，而没有任何组织的组织可能仅要求任何成员都可以签名。

13. 示例：渠道配置策略

了解策略始于检查 configtx.yaml 通道策略的定义位置。我们可以 configtx.yaml 在 BYFN (第一网络) 教程中使用该文件来查看两种策略语法类型的示例。导航到 fabric-samples / first-network 目录，并检查 configtx.yaml 文件中的 BYFN。

该文件的第一部分定义了网络的组织。每个组织定义中都有该组织的默认策略，尽管您可以根据需要命名策略。每个策略都有一个描述策略（或）表达方式的和。

Readers, Writers, Admins, and EndorsementTypeSignatureImplicitMetaRule

下面的 BYFN 示例显示了 Org1 系统通道中的组织定义，其中策略 Type 是 Signature，并且认可策略规则定义为 "OR('Org1MSP.peer')"，这意味着 Org1MSP 需要成为成员的对等方进行签名。这些签名策略成为 ImplicitMeta 策略指向的子策略。

单击此处查看使用签名策略定义的组织的示例

下面的例子演示了 ImplicitMeta 在使用策略类型 Orderer 的部分 configtx.yaml 文件，该文件定义了订货的默认行为，并且还包含了相关的政策 Readers, Writers 和 Admins。同样，这些 ImplicitMeta 策略是根据我们在上面的代码段中看到的其潜在的 Signature 子策略进行评估的。

单击此处查看 ImplicitMeta 策略的示例

14. Fabric 链代码生命周期

在 Fabric Alpha 2.0 版本中，引入了新的链码生命周期过程，从而使用更加民主的过程来管理网络上的链码。新过程使多个组织可以对链码的使用方式进行投票，然后才能在通道上使用它。这很重要，因为正是这个新生命周期过程与该过程中指定的策略的组合决定了整个网络的安全性。有关该流程的更多详细信息，请参见《运营商 Chaincode》教程，但出于本主题的目的，您应该了解如何在该流程中使用策略。新流程包括指定策略的两个步骤：链码何时由组织成员批准以及何时提交到频道。

该文件的 Application 部分 configtx.yaml 包括默认的链码生命周期认可策略。在生产环境中，您将针对自己的用例自定义此定义。

```
#####
#
# SECTION: Application
#
# - This section defines the values to encode into a config transaction or
# genesis block for application related parameters
#
#####
Application: &ApplicationDefaults
```

```

# Organizations is the list of orgs which are defined as participants on
# the application side of the network
Organizations:

# Policies defines the set of policies at this Level of the config tree
# For Application policies, their canonical path is
# /Channel/Application/<PolicyName>

Policies:

Readers:
  Type: ImplicitMeta
  Rule: "ANY Readers"

Writers:
  Type: ImplicitMeta
  Rule: "ANY Writers"

Admins:
  Type: ImplicitMeta
  Rule: "MAJORITY Admins"

LifecycleEndorsement:
  Type: ImplicitMeta
  Rule: "MAJORITY Endorsement"

Endorsement:
  Type: ImplicitMeta
  Rule: "MAJORITY Endorsement"

```

- 该 `LifecycleEndorsement` 策略控制谁需要批准 `chaincode` 定义。
- 该 `Endorsement` 策略是链码的默认背书策略。在下面的更多内容。

15. 链码认可政策

在使用 Fabric 链码生命周期批准链码并将其提交给渠道后，便为链码指定了背书策略（即，一种背书策略涵盖了与链码相关联的所有状态）。可以通过参考在通道配置中定义的认可策略来指定认可策略，也可以通过显式指定签名策略来指定认可策略。

如果在批准步骤中未明确指定背书策略，则使用默认 `Endorsement` 策略，这意味着属于不同渠道成员（组织）的大多数对等方需要针对链码执行和验证交易，以便进行交易被认为是有效的。此默认策略允许加入渠道的组织自动添加到链码认可策略中。如果您不想使用默认的背书策略，请使用“签名”策略格式来指定更复杂的背书策略（例如，要求链码由一个组织，然后由渠道上的其他组织之一背书）。`"MAJORITY Endorsement"`

签名策略还允许您包括在内 `principals`，这仅仅是将身份与角色进行匹配的一种方式。委托人就像用户 ID 或组 ID，但是它们用途更广，因为它们可以包含参与者身份的各种属性，例如参与者的组织，组织单位，角色，甚至参与者的特定身份。当我们谈论主体时，它们是确定其权限的属性。主体被描述为“MSP.ROLE”，其中 `MSP` 代表所需的 MSP ID（组织），并 `ROLE` 代表四个接受的角色之一：成员，管理员，客户和对等。当用户向 CA 注册时，角色与身份相关联。您可以自定义 Fabric CA 上可用的角色列表。

有效主体的一些示例是：

- 'Org0.Admin': Org0 MSP 的管理员
- 'Org1.Member': Org1 MSP 的成员

- 'Org1.Client': Org1 MSP 的客户端
- 'Org1.Peer': Org1 MSP 的同级
- 'OrdererOrg.Orderer': OrdererOrg MSP 中的排序节点

在某些情况下，特定状态（换句话说，特定键值对）可能需要具有不同的认可策略。这种**基于状态的认可**允许默认的链码级认可策略被指定密钥的不同策略所覆盖。

有关如何编写背书策略的更多信息，请参阅《操作指南》中有关背书策略的主题。

注意：根据您使用的 Fabric 版本，策略的工作方式有所不同：

在 2.0 Alpha 发行版之前的 Fabric 发行版中，可以在链码实例化期间或使用 chaincode 生命周期命令来更新链码认可策略。如果在实例化时未指定，则背书策略默认为“渠道中组织的任何成员”。例如，具有“Org1”和“Org2”的频道的默认背书策略为“OR ('Org1.member', 'Org2.member') ”。

从 Alpha 2.0 版本开始，Fabric 引入了新的链码生命周期过程，该过程使多个组织可以就链码在通道上使用之前如何操作达成一致。新流程要求组织同意定义链码的参数，例如名称，版本和链码认可策略。

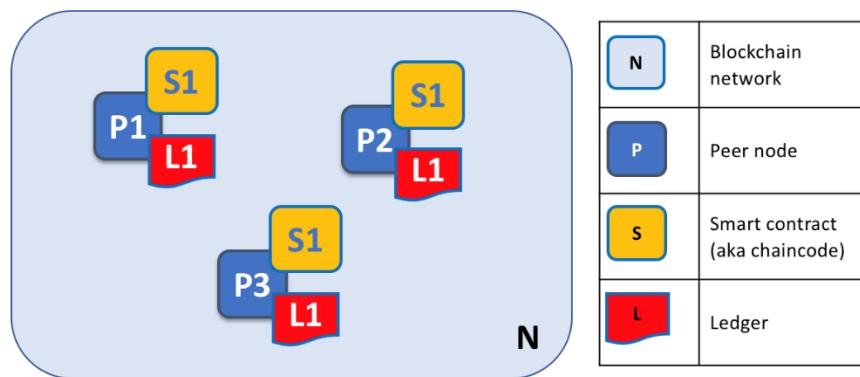
16. 覆盖策略定义

Hyperledger Fabric 包含默认策略，这些策略对于开始，开发和测试区块链非常有用，但它们应在生产环境中进行自定义。您应该了解 `configtx.yaml` 文件中的默认策略。信道配置策略可以任意动词进行扩展，除了默认的在。当您通过编辑排序节点系统通道或特定通道的覆盖默认策略时，通过发布配置更新来覆盖排序节点系统和应用程序通道。`Readers`, `Writers`, `Admins``configtx.yaml``configtx.yaml`
请参阅有关更新通道配置的主题以 获取更多信息。

对等节点

区块链网络主要由一组对等节点（或简称为对等）组成。对等是网络的基本元素，因为它们托管分类帐和智能合约。回想一下，分类帐一成不变地记录了智能合约生成的所有交易（在 Hyperledger Fabric 中包含在链式代码中，稍后会详细介绍）。智能合约和分类帐分别用于封装网络中的共享过程和共享信息。对等方的这些方面使它们成为了解 Fabric 网络的良好起点。

区块链网络的其他元素当然很重要：分类帐和智能合约，排序者，策略，渠道，应用程序，组织，身份和成员身份，您可以在其专用部分中阅读有关它们的更多信息。本节重点介绍对等点及其与 Fabric 网络中其他元素的关系。



区块链网络由对等节点组成，每个对等节点可以保存分类账的副本和智能合约的副本。在此示例中，网络 N 由对等端 P_1 , P_2 和 P_3 组成，每个对端 P_1 , P_2 和 P_3 维护自己的分布式账本 L_1 实例。 P_1 , P_2 和 P_3 使用相同的链码 S_1 来访问其分布式账本的副本。

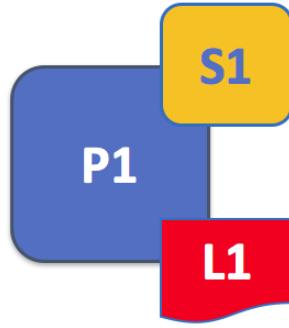
可以创建，启动，停止，重新配置甚至删除对等体。它们公开了一组 API，使管理员和应用程序可以与其提供的服务进行交互。我们将在本节中了解有关这些服务的更多信息。

1. 术语一词

Fabric 通过称为**链码**的技术概念来 实现**智能合约** -只是访问**账本**的一段代码，以一种受支持的编程语言编写。在本主题中，我们通常使用术语 **chaincode**，但是如果您更习惯该术语，可以随时将其作为**智能合约**来阅读。这是同一件事！如果您想了解有关链码和智能合约的更多信息，请查看我们[有关智能合约和链码的文档](#)。

2. 分类帐和 Chaincode

让我们更详细地介绍一个对等节点。我们可以看到，同时承载分类帐和链码的是对等方。更准确地，对等体实际上承载 实例 的分类账的，并且实例 chaincode 的。请注意，这在光纤网络中提供了有意的冗余-避免了单点故障。在本节后面，我们将了解有关区块链网络的分布式和分散式性质的更多信息。

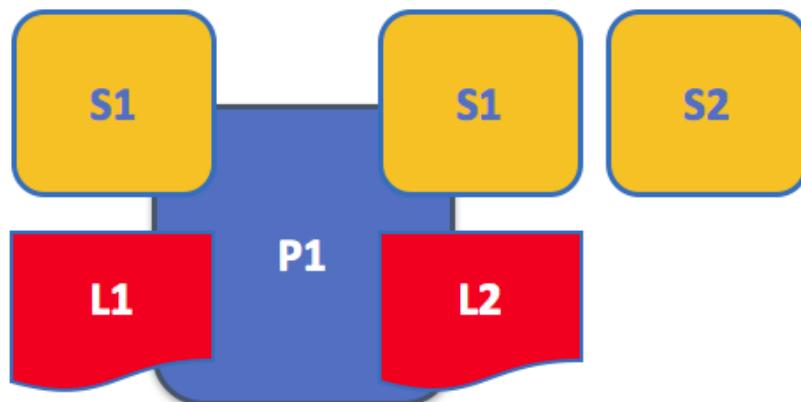


对等方托管分类账实例和链码实例。在此示例中，**P1** 承载分类帐 **L1** 的实例和链码 **S1** 的实例。单个对等主机上可以托管许多分类帐和链码。

由于对等方是分类帐和链码的宿主，因此应用程序和管理员如果要访问这些资源，必须与对等方进行交互。这就是为什么将对等方视为 Fabric 网络最基本的构建基块的原因。首次创建同位体时，既没有分类帐，也没有链码。稍后我们将看到在同级上如何创建分类帐以及如何安装链码。

3. 多个分类帐

对等节点可以承载多个分类帐，这很有用，因为它允许灵活的系统设计。最简单的配置是让对等方管理一个分类帐，但是对等方在需要时托管两个或多个分类帐绝对是合适的。

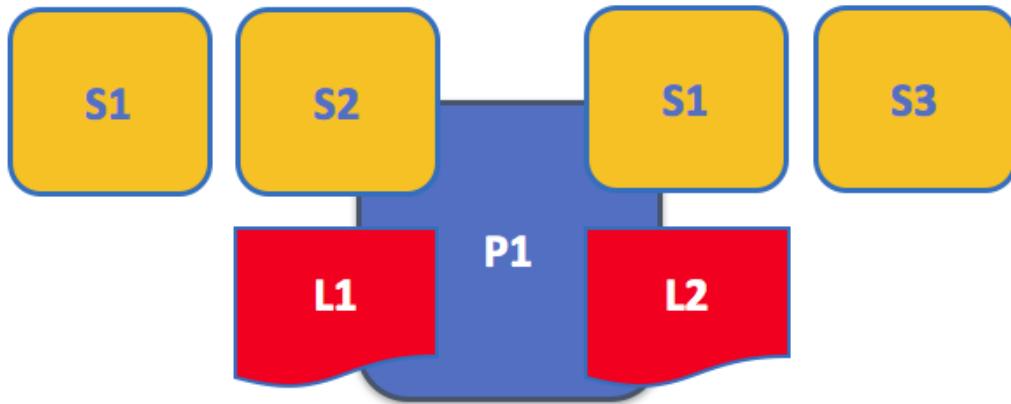


承载多个分类帐的对等体。对等主机托管一个或多个分类帐，每个分类帐具有适用于它们的零个或多个链码。在此示例中，我们可以看到对等 **P1** 承载分类帐 **L1** 和 **L2**。使用链码 **S1** 访问分类帐 **L1**。另一方面，可以使用链码 **S1** 和 **S2** 访问分类帐 **L2**。

尽管对等点完全有可能托管一个账本实例，而不托管任何访问该账本的链码，但是很少有对等端以这种方式配置。绝大多数对等点将至少安装一个链码，可以查询或更新对等点的分类帐实例。值得一提的是，无论用户是否已安装供外部应用程序使用的链码，对等方也始终具有特殊的**系统链码**。在本主题中将不对这些进行详细讨论。

4. 多个链码

对等方拥有的分类账数量与可以访问该分类账的链码数量之间没有固定的关系。一个对等方可能有许多可用的链码和分类帐。



托管多个链码的同位体的示例。每个分类帐可以有许多访问它的链码。在此示例中，我们可以看到对等 P_1 承载分类帐 L_1 和 L_2 ，其中 L_1 由链码 S_1 和 S_2 访问，而 L_2 由 S_1 和 S_3 访问。我们可以看到 S_1 可以访问 L_1 和 L_2 。

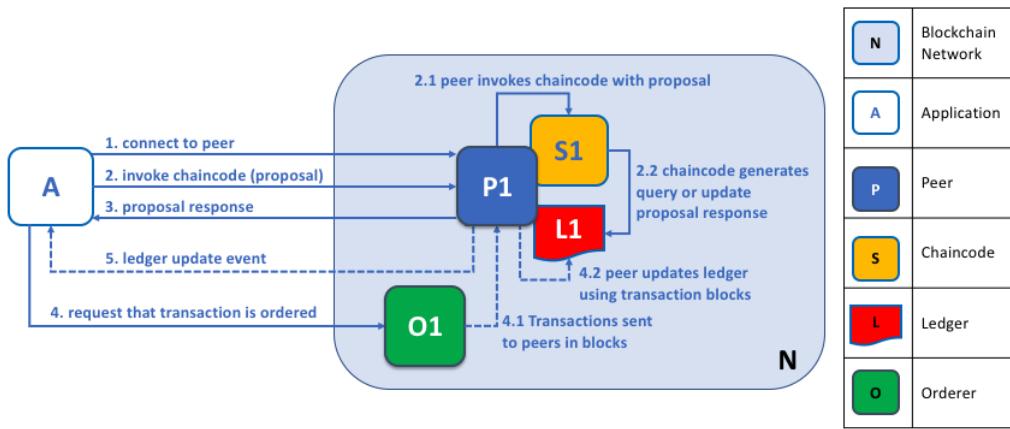
稍后我们将看到为什么在对等主机上托管多个分类帐或多个链码时，Fabric 中的通道概念如此重要的原因。

5. 应用程序和对等

现在，我们将展示应用程序如何与对等方交互以访问分类帐。分类账查询交互包括应用程序和对等方之间的简单三步对话；分类帐与更新的交互要复杂得多，并且需要两个额外的步骤。我们对这些步骤进行了一些简化，以帮助您开始使用 Fabric，但是不用担心-最重要的是要了解分类账查询与分类账更新事务样式之间的应用程序对等交互。

当应用程序需要访问分类帐和链码时，它们总是连接到同级。Fabric 软件开发套件 (SDK) 使程序员可以轻松实现它-它的 API 使应用程序可以连接到同级，调用链代码以生成事务，将事务提交到网络以进行排序，验证和提交给分布式分类帐，并接收事件此过程完成后。

通过对等连接，应用程序可以执行链码来查询或更新分类帐。分类帐查询事务的结果将立即返回，而分类帐更新涉及应用程序，对等方和排序节点之间更复杂的交互。让我们对此进行更详细的研究。



对等与排序节点一起确保每个分类账都保持最新。在此示例中，应用程序 A 连接到 P1 并调用链码 S1 以查询或更新分类帐 L1。P1 调用 S1 以生成包含查询结果或提议的分类帐更新的提议响应。应用程序 A 收到提案响应，并且对于查询，该过程现在已完成。为了进行更新，A 根据所有响应构建一个事务，并将其发送给 O1 进行订购。O1 将整个网络中的交易收集到多个区块中，并将其分配给所有对等节点，包括 P1。P1 在提交给 L1 之前先验证交易。一旦 L1 更新，P1 就会生成一个事件，该事件由 A 接收，表示完成。

对等方可以立即将查询结果返回给应用程序，因为满足查询所需的所有信息都在对等方的总账本地副本中。对等方从不与其他对等方协商以响应来自应用程序的查询。但是，应用程序可以连接到一个或多个对等端以发出查询。例如，要确认多个对等方之间的结果，或者如果怀疑信息可能已过时，则从另一个对等方检索最新结果。在该图中，您可以看到分类帐查询是一个简单的三步过程。

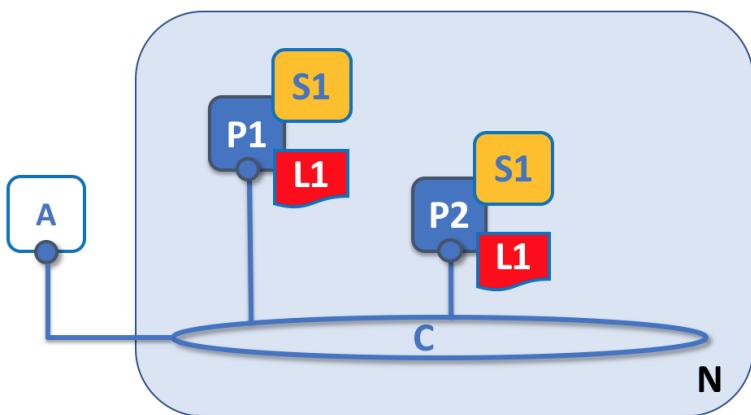
更新事务以与查询事务相同的方式启动，但有两个额外的步骤。尽管更新账本的应用程序还连接到对等点以调用链码，但与账本查询的应用程序不同，但单个对等点此时无法执行账本更新，因为其他对等点必须首先同意该更改，此过程称为**共识**。因此，对等方向应用程序返回了**建议的更新**-该对等方将在其他对等方事先同意的情况下应用该更新。第一个额外的步骤（第四步）要求应用程序将一组适当的匹配的提议更新集发送到整个对等网络，以作为对各自账本的承诺的交易。这是通过应用程序使用**排序**程序来实现的将交易打包成块，然后将其分发到整个对等网络，在将它们应用于每个对等的本地分类帐副本之前，可以在其中进行验证。由于整个订购过程需要一些时间（几秒钟）才能完成，因此将异步通知应用程序，如步骤 5 所示。

在本节的后面，您将了解有关此订购过程的详细性质的更多信息-有关此过程的真正详细信息，请参见[交易流](#)主题。

6. 对等节点和渠道

尽管本节的重点是对等点而不是渠道，但值得花一些时间来了解对等点如何通过渠道与彼此以及与应用程序交互——一种机制，通过这种机制，区块链网络中的一组组件可以进行私下通信和交易。

这些组件通常是对等节点，排序者节点和应用程序，并且通过加入渠道，它们同意协作以共同共享和管理与该渠道关联的分类帐的相同副本。从概念上讲，您可以将频道视为与朋友组相似（尽管频道的成员当然不需要成为朋友！）。一个人可能有几组朋友，每组都有他们一起做的活动。这些小组可能是完全独立的（一群工作朋友，而不是一群爱好朋友），或者它们之间可能会有一些交叉。但是，每个组都是其自己的实体，具有某种“规则”。



N	Blockchain Network	L	Ledger
C	Channel	A	Application
P	Peer	PA	Principal PA (e.g. A, P1) communicates via channel C.
S	Chaincode		

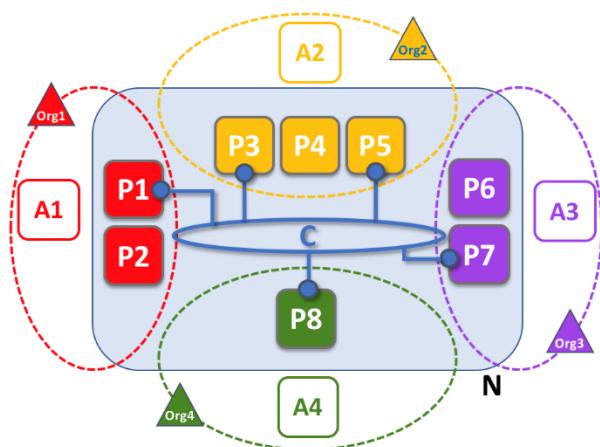
通道允许一组特定的对等方和应用程序在区块链网络内相互通信。在此示例中，应用程序 **A** 可以使用通道 **C** 直接与同级 **P1** 和 **P2** 通信。您可以将通道视为特定应用程序和同级之间进行通信的路径。（为简单起见，排序节点未在此图中显示，但必须存在于正常运行的网络中。）

我们看到通道并不像对等体那样存在-将通道视为由物理对等体集合形成的逻辑结构是更合适的。理解这一点至关重要-对等方提供访问和管理通道的控制点。

7. 对等节点和组织

现在您已经了解了同级及其与分类帐，链码和渠道的关系，您将能够看到多个组织如何一起形成一个区块链网络。

区块链网络由组织的集合而不是单个组织管理。对等对于这种分布式网络的构建至关重要，因为对等网络由这些组织拥有，并且是这些组织的网络连接点。



N	Blockchain Network	L	Ledger
C	Channel	A	Application
P	Peer	PA	Principal PA (e.g. A1, P5) communicates via channel C.
		Org	Organization
Org R owns application A1 and peers P1, P2.			

具有多个组织的区块链网络中的对等点。区块链网络是由不同组织拥有和贡献的对等体建立的。在此示例中，我们看到四个组织贡献了八个对等方组成一个网络。通道 **C** 连接网络 **N** 中的 **P1, P3, P5, P7** 和 **P8** 中的五个对等方。这些组织拥有的其他对等方尚未加入此渠道，但通常已加入至少一个其他渠

道。由特定组织开发的应用程序将连接到其各自组织的对等方以及不同组织的对等方。同样，为简单起见，该图中未显示排序者节点。

看到区块链网络的形成过程非常重要，这一点非常重要。网络由向其贡献资源的多个组织组成和管理。对等是我们在本主题中讨论的资源，但是组织提供的资源不仅仅是对等。这里有一个原则在起作用-如果组织没有将各自的资源投入到集体网络中，那么网络实际上就不存在。而且，网络随着这些协作组织提供的资源而增长和收缩。

您可以看到（除了排序服务之外）没有集中式资源-在上面的示例中，如果组织不提供其对等项，则网络 N 将不存在。这反映了这样一个事实：除非有组织提供构成该网络的资源，否则该网络不存在任何有意义的意义。此外，网络不依赖于任何单个组织，只要一个组织存在，它就会继续存在，无论其他组织可能来去去。这是网络去中心化意味着什么的核心。

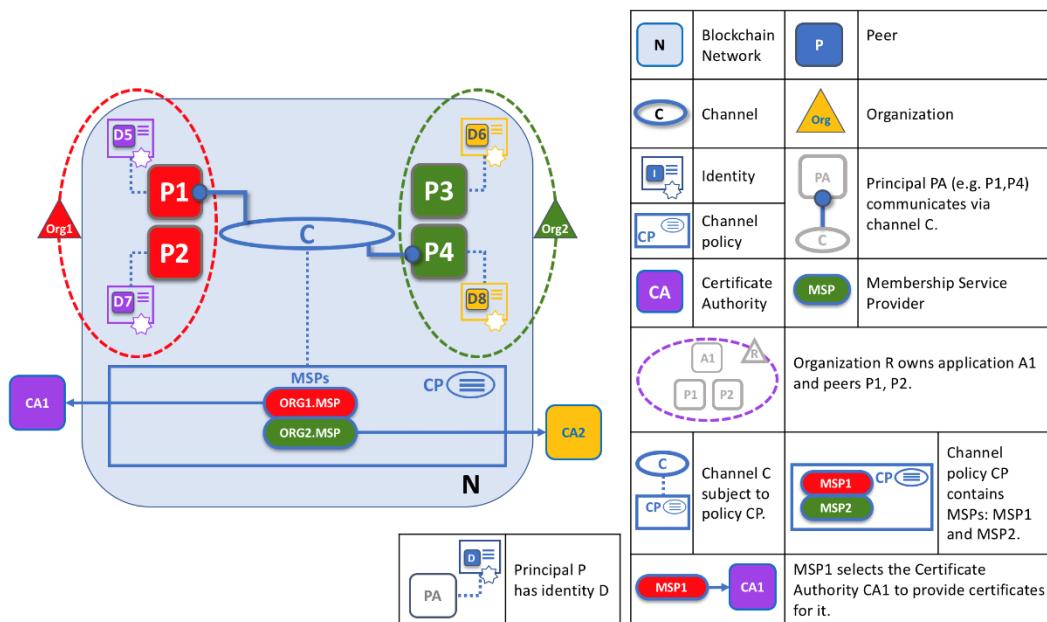
如上例所示，不同组织中的应用程序可能相同，也可能不同。那是因为，组织的应用程序如何处理其对等人的分类帐副本完全取决于组织。这意味着应用程序和表示逻辑在组织之间可能会有所不同，即使它们各自的对等节点托管的账本数据完全相同。

应用程序可以连接到其组织中的对等方，也可以连接到另一个组织中的对等方，这取决于所需的账本交互的性质。对于分类账查询交互，应用程序通常连接到其自身组织的对等方。对于分类账更新交互，我们将在后面看到为什么应用程序需要连接到代表批准分类账更新所需的每个组织的同级。

8. 对等与身份

既然您已经了解了来自不同组织的同龄人如何形成一个区块链网络，那么值得花一些时间来了解如何将其同龄人由其管理员分配给组织。

对等方具有通过特定证书颁发机构通过数字证书分配给他们的身份。您可以在本指南的其他地方阅读有关 X.509 数字证书如何工作的更多信息，但就目前而言，数字证书就像是 ID 卡，可以提供有关对等方的大量可验证信息。网络中的每个对等方都由其所属组织的管理员分配了数字证书。



当对等方连接到通道时，其数字证书通过通道 MSP 标识其拥有的组织。在此示例中，P1 和 P2 具有 CA1 颁发的身份。通道 C 根据其通道配置中的策略，确定应该使用 ORG1. MSP 将来自 CA1 的标识与 Org1 关联。同样，P3 和 P4 被 ORG2. MSP 标识为 Org2 的一部分。

每当对等方使用通道连接到区块链网络时，通道配置中的策略都会使用对等方的身份来确定其权限。身份到组

织的映射由称为成员资格服务提供者的组件提供 (MSP) - 它确定如何将对等方分配给特定组织中的特定角色，并因此获得对区块链资源的适当访问。此外，对等方只能由单个组织拥有，因此与单个 MSP 关联。在本节的后面，我们将学习更多有关对等访问控制的知识，并且在本指南的其他地方，将有一个完整的章节介绍了 MSP 和访问控制策略。但就目前而言，可以将 MSP 视为在区块链网络中的个人身份和特定组织角色之间提供链接。

暂时，对等以及与区块链网络交互的所有事物都从其数字证书和 MSP 获得其组织身份。对等方，应用程序，最终用户，管理员和排序节点如果要与区块链网络进行交互，则必须具有标识和关联的 MSP。我们为使用身份（委托人）与区块链网络交互的每个实体命名。您可以在本指南中的其他地方了解更多有关校长和组织的信息，但是到目前为止，您所了解的知识还远远不够，可以继续理解对等节点！

最后，请注意，对等实体的物理位置并不重要-它可以位于云中，也可以位于组织之一拥有的数据中心中，或者位于本地计算机上-与它相关联的数字证书可以识别它由特定组织拥有。在上面的示例中，P3 可以托管在 Org1 的数据中心中，但是只要与它关联的数字证书由 CA2 颁发，那么它就由 Org2 拥有。

9. 对等节点和排序节点

我们已经看到，对等点构成了区块链网络的基础，托管了分类账和智能合约，可以由对等连接的应用程序查询和更新。但是，应用程序和对等方之间进行交互以确保每个对等方的分类账保持一致的机制是由称为 *orderer* 的特殊节点介导的，现在我们将这些注意力转向这些节点。

更新事务与查询事务完全不同，因为单个对等方自身无法更新分类帐-更新需要网络中其他对等方的同意。对等方需要网络中的其他对等方批准分类帐更新，然后才能将其应用于对等方的本地分类帐。此过程称为“共识”，它比简单的查询需要更长的时间才能完成。但是，当所有需要批准交易的对等方都批准了该交易并将交易提交到分类帐时，对等方将通知其连接的应用程序分类帐已更新。在本部分中，您将获得有关同级和排序节点如何管理共识过程的更多详细信息。

具体来说，想要更新分类帐的应用程序涉及 3 个阶段的过程，这确保了区块链网络中的所有对等方都保持其分类帐彼此一致。

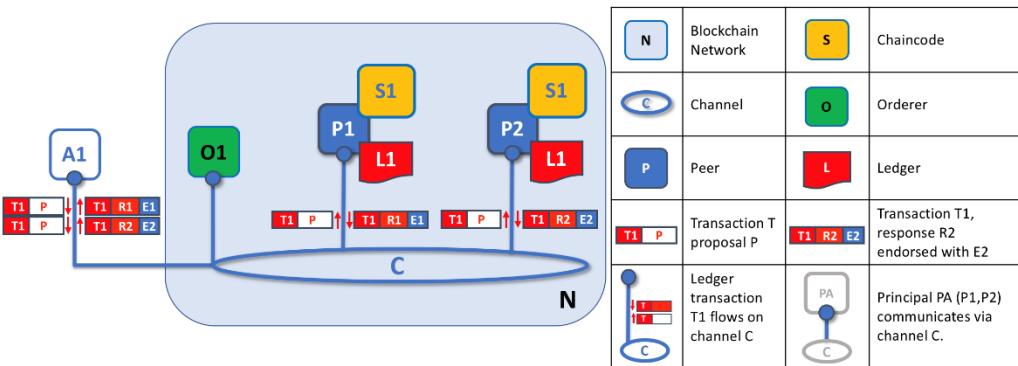
- 在第一阶段，应用程序与背书对等方的子集一起工作，每个背书对等方都向应用程序提供对提议的分类账更新的背书，但不对提议的分类账副本应用其提议的更新。
- 在第二阶段，将这些单独的背书作为交易收集在一起，并打包成块。
- 在第三阶段（也是最后一个阶段）中，将这些块分配回每个对等方，在对每个对等方进行验证之后，才将每个事务提交给该对等方的分类帐副本。

正如您将看到的，排序节点是此过程的核心，因此让我们更详细地研究应用程序和对等方如何使用排序节点来生成分类帐更新，这些更新可以一致地应用于分布式复制分类帐。

10. 阶段 1：提案

交易工作流程的第一阶段涉及应用程序和一组对等方之间的交互-它不涉及排序节点。第一阶段仅与一个要求不同组织的认可对等节点同意提议的链码调用结果的应用程序有关。

从第 1 阶段开始，应用程序会生成一个交易建议，然后将其发送给每个必需的对等方组以进行认可。然后，这些背书对等方中的每一个都使用交易建议独立执行链码以生成交易建议响应。它不会将此更新应用于分类帐，而只是对其进行签名并将其返回给应用程序。一旦应用程序收到了足够数量的已签名提案响应，交易流程的第一阶段便完成了。让我们更详细地研究这个阶段。



交易建议书由返回认可的建议书响应的对等节点独立执行。在此示例中，应用程序 A1 生成事务 T1 提议 P，并将其发送到通道 C 上的对等方 P1 和对等方 P2。P1 使用事务 T1 提议 P 生成事务 T1 响应 R1 并由 E1 认可，从而执行 S1。P2 独立地使用事务 T1 提议 P 执行 S1，并生成事务 T1 响应 R2，并由 E2 认可。应用程序 A1 收到两个对事务 T1 的认可响应，即 E1 和 E2。

最初，应用程序选择一组对等点以生成一组建议的分类账更新。应用程序选择了哪些对等方？好吧，这取决于背书策略(为链码定义)，背书策略定义了需要在网络接受之前对提议的分类账更改进行背书的组织集合。从字面上看，这就是达成共识的意思-每个重要组织都必须已经批准了建议的分类帐更改，然后才能将其接受到任何同级分类帐中。

对等节点通过添加其数字签名并使用其私钥对整个有效负载进行签名来认可提案响应。这种认可可以随后用于证明该组织的同级产生了特定的响应。在我们的示例中，如果对等方 P1 由组织 Org1 拥有，则批注 E1 对应于一个数字证明，即“由 Org1 的对等方 P1 提供了账本 L1 上的交易 T1 响应 R1！”。

当应用程序收到足够的对等方签署的提案响应时，阶段 1 结束。我们注意到，对于同一个交易建议，不同的同位体可以对应用程序返回不同的，因此不一致的交易响应。可能只是结果是在不同时间使用不同状态的分类帐在不同的同位体上生成的，在这种情况下，应用程序可以简单地请求更新的投标响应。结果的可能性较小，但更严重的是，结果可能会有所不同，因为链码是不确定的。非确定性是链码和分类账的敌人，如果发生，则表明拟议的交易存在严重问题，因为显然不能将不一致的结果应用于分类账。单个对等方无法知道他们的交易结果是不确定的，因此必须先收集交易响应以进行比较，然后才能检测到不确定性。(严格来说，这还不够，但是我们将讨论推迟到事务部分，在此部分将详细讨论不确定性。)

在阶段 1 结束时，应用程序可以随意丢弃不一致的事务响应，从而有效地尽早终止事务工作流。稍后我们将看到，如果应用程序尝试使用一组不一致的事务响应来更新分类帐，它将被拒绝。

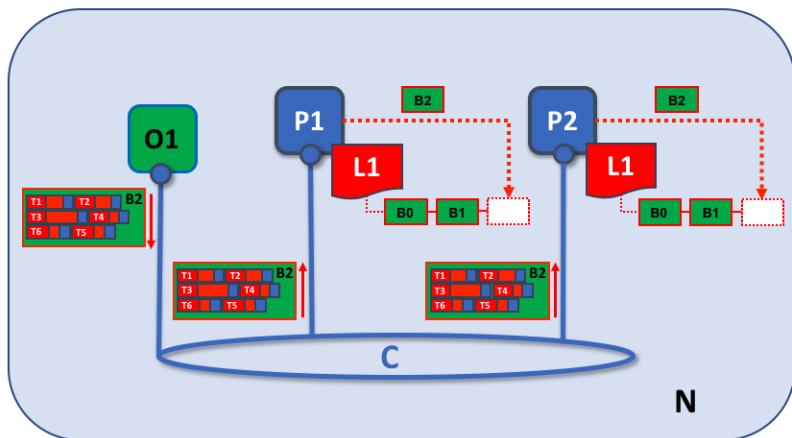
11. 阶段 2：将交易排序和打包成块

交易工作流程的第二阶段是打包阶段。排序节点对于此过程至关重要-它从许多应用程序接收包含认可的交易建议响应的交易，并将交易定为区块。有关订购和包装阶段的更多详细信息，请查看 [有关排序阶段的概念性信息](#)。

12. 阶段 3：验证和提交

在第 2 阶段结束时，我们看到排序节点负责简单但至关重要的过程，这些过程包括收集提议的交易更新，进行订购并将它们打包成块，以便分发给对等方。

交易工作流程的最后阶段涉及从排序节点到对等方的块的分发和后续验证，在这里可以将它们提交到分类账。具体来说，在每个对等方，对一个区块中的每个交易都进行验证，以确保在提交到分类账之前，所有相关组织都一致认可该交易。失败的交易将保留以进行审核，但不会提交到分类账。



N	Blockchain Network	P	Peer
C	Channel	O	Orderer
L	Ledger	B	Block B
L1	Ledger L1 has blockchain with blocks B0, B1	T1, T2, T3...	Block B1 contains transactions T1, T2, T3...
B1	Block B1 flows on channel C	PA	Principal PA (P1, P2) communicates via channel C.

排序节点节点的第二个作用是将块分配给同级。在该示例中，定购者 O1 将块 B2 分配给对等体 P1 和对等体 P2。对等体 P1 处理块 B2，从而将新块添加到 P1 上的分类帐 L1。并行地，对等点 P2 处理块 B2，导致将新块添加到 P2 上的分类帐 L1。一旦该过程完成，分类帐 L1 就已在对等端 P1 和 P2 上进行了一致更新，并且每个分类帐可以通知连接的应用程序交易已处理。

阶段 3 从排序节点将块分配给与其连接的所有对等点开始。对等点通过通道连接到排序节点，以便在生成新块时，将向连接到排序节点的所有对等点发送新块的副本。每个对等方将独立处理此块，但方法与通道上的每个其他对等方完全相同。这样，我们将看到分类帐可以保持一致。还值得注意的是，并非每个对等点都需要连接到排序节点，对等点可以使用八卦 Gossip 协议将块级联到其他对等点，后者也可以独立处理它们。但是，让我们将讨论留给其他时间！

收到块后，对等方将按照它出现在块中的顺序处理每个事务。对于每笔交易，每个对等方都将根据签注策略验证交易是否已由所需组织 批准生成交易的链码的数量。例如，某些交易可能只需要由单个组织认可，而其他交易可能需要多次认可才能被视为有效。此验证过程将验证所有相关组织均已产生相同的结果或结果。还要注意，此验证与阶段 1 中的背书检查不同，在阶段 1 中，应用是从背书对等方接收响应并做出发送提案交易的决定。如果应用程序通过发送错误的交易违反了背书策略，则对等方仍然可以在阶段 3 的验证过程中拒绝交易。

如果交易已正确认可，则对等方将尝试将其应用于分类帐。为此，对等方必须执行分类帐一致性检查，以验证生成建议更新时分类帐的当前状态与分类帐的状态兼容。即使交易已得到完全认可，这也不总是可能的。例如，另一笔交易可能已更新分类帐中的同一资产，因此该交易更新不再有效，因此无法再应用。这样，每个对等方的分类帐副本在整个网络中保持一致，因为他们每个人都遵循相同的验证规则。

对等方成功验证了每个单独的交易后，它将更新分类帐。失败的事务不应用于分类帐，但保留它们以进行审核，与成功的事务一样。这意味着对等块几乎与从定单接收到的块完全相同，除了该块中每个事务的有效或无效指示符。

我们还注意到，第 3 阶段不需要运行链码-仅在第 1 阶段才完成，这很重要。这意味着链代码仅在认可节点上可用，而不是在整个区块链网络上可用。这通常很有用，因为它可以使链码的逻辑对认可组织保密。这与链码的输出（交易建议响应）相反，链码的输出与渠道中的每个对等方共享，无论他们是否认可交易。认可对等方的这种专业化旨在帮助实现可伸缩性和机密性。

最后，每次将一个区块提交给对等方的分类帐时，该对等方都会生成一个适当的事件。区块事件包括完整的区块内容，而区块交易事件仅包含摘要信息，例如区块中的每个交易是否已验证或无效。链码执行产生的链码事件也可以在此时发布。应用程序可以注册这些事件类型，以便在事件发生时得到通知。这些通知结束了事务工作流的第三阶段和最后阶段。

总而言之，第 3 阶段将看到由定购者生成的块始终应用于分类帐。严格按顺序将交易划分为块，每个对等方都可以验证交易更新是否在整个区块链网络中得到一致应用。

13. 排序节点和共识

整个交易流程的整个过程称为共识，因为在排序节点的调解下，所有对等方都已就交易的顺序和内容达成了共识。共识是一个多步骤的过程，仅当过程完成时才通知应用程序分类帐更新，这可能在不同的对等点发生的时间略有不同。

我们将在未来的排序节点主题中更详细地讨论排序节点，但现在，将排序节点视为从应用程序收集和分发提议的分类帐更新以供对等方验证并包含在分类帐中的节点。

现在，我们已经完成了对对等节点及其与 Fabric 相关的其他组件的浏览。我们已经看到，在许多方面，对等方是最基本的元素-它们形成网络，主机链代码和分类帐，处理交易建议和响应，并通过始终向其应用交易更新来使分类帐保持最新。

智能合约和链码

受众：架构师，应用程序和智能合约开发人员，管理员

从应用程序开发人员的角度来看，智能合约与分类帐一起构成了 Hyperledger Fabric 区块链系统的核心。账本保存有关一组业务对象的当前和历史状态的事实，而智能合约定义可执行逻辑，生成生成添加到账本的新事实的可执行逻辑。一个 chaincode 通常由管理员进行部署组相关的智能合同，但也可用于低级别的系统编程。在本主题中，我们将重点介绍为什么智能合约和链码同时存在以及如何以及何时使用它们。

在本主题中，我们将介绍：

- 什么是智能合约
- 术语说明
- 智能合约和分类帐
- 如何制定智能合约
- 认可政策的重要性
- 有效交易
- 通道和链码定义
- 智能合约之间的沟通
- 什么是系统链码？

1. 智能合约

在业务彼此之间进行交易之前，他们必须定义一组通用合同，涵盖通用条款，数据，规则，概念定义和流程。这些合同放在一起，构成了控制交易双方之间所有交互的业务模型。



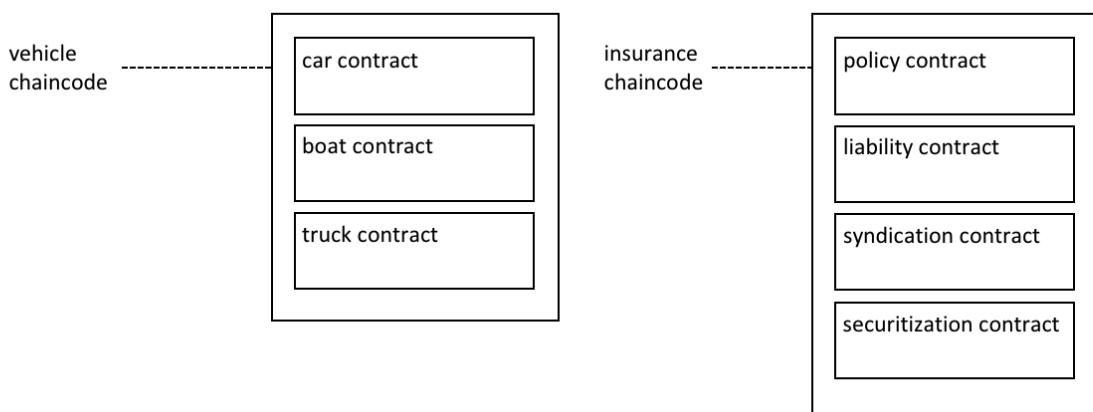
智能合约以可执行代码定义了不同组织之间的规则。应用程序调用智能合约以生成记录在分类账上的交易。

使用区块链网络，我们可以将这些合同转换为可执行程序（在业界称为**智能合同**），以开辟各种新的可能性。这是因为智能合约可以为**任何**类型的业务对象实现治理规则，以便在执行智能合约时可以自动执行这些规则。例如，智能合约可以确保在指定的时间范围内完成新车交付，或者确保按照预先安排的条款释放资金，从而分别改善了货物或资金的流动。但是，最重要的是，智能合同的执行比人工人工业务流程要高效得多。

在上图中，我们可以看到这两个组织，`ORG1` 并 `ORG2` 定义了一个 `car` 聪明的合同 `query`, `transfer` 和 `update` 汽车。这些组织的应用程序调用此智能合约在业务流程中执行约定的步骤，例如将特定汽车的所有权从转让 `ORG1` 给 `ORG2`。

2. 术语

Hyperledger 面料用户经常使用的术语**智能合同**和 **chaincode** 互换。通常，智能合约定义了控制世界状态中包含的业务对象生命周期的**交易逻辑**。然后将其打包成链码，然后将其部署到区块链网络。可以将智能合约视为支配事务，而链码则可以控制如何打包智能合约以进行部署。



智能合约在链码中定义。可以在同一链码中定义多个智能合约。部署链码后，其中的所有智能合约都可用于应用程序。

在图中，我们可以看到一个 `vehicle` 包含三个聪明的合同 chaincode: `cars`, `boats` 和 `trucks`。我们还可以看到一个 `insurance` 包含四个智能合同 chaincode: `policy` 和 `liability`、`syndication` 和 `securitization`。在这两种情况下，这些合同均涉及与车辆和保险有关的业务流程的关键方面。在本主题中，我们将以合同为例。我们可以看到，智能合约是与特定业务流程相关的领域特定程序，而链码是一组相关智能合约的技术容器。

3. 分类帐

在最简单的层次上，区块链一成不变地记录更新账本中状态的交易。智能合约以编程方式访问总账的两个不同部分：一个**区块链**（不可变地记录所有交易的历史记录）和一个**世界状态**，其中保存着这些状态的当前值，因为它是对象的当前值。通常需要。

智能合约主要在世界状态下**放置**、**获取**和**删除**状态，也可以查询不可变的区块链交易记录。

- 一个 **GET** 通常代表一个查询，检索有关业务对象的当前状态信息。
- 一个**put** 通常会创建一个新的业务对象或修改现有的分类账状态。
- 一个**delete** 典型代表去除总账的当前状态的业务对象，而不是它的历史。

智能合约有许多可用的 API。至关重要的是，在所有情况下，无论交易是在世界状态下创建，读取，更新

还是删除业务对象，区块链都包含这些更改的不可变记录。

4. 发展历程

智能合约是应用程序开发的重点，正如我们已经看到的，可以在单个链码中定义一个或多个智能合约。将 Chaincode 部署到网络后，该网络中的所有组织均可使用其所有智能合约。这意味着只有管理员才需要担心链码。其他人都可以根据智能合约进行思考。

智能合约的核心是一组 `transaction` 定义。例如，查看 `fabcar.js`，您可以在 其中看到创建新车的智能合约交易：

```
async createCar(ctx, carNumber, make, model, color, owner) {
```

```
    const car = {
        color,
        docType: 'car',
        make,
        model,
        owner,
    };

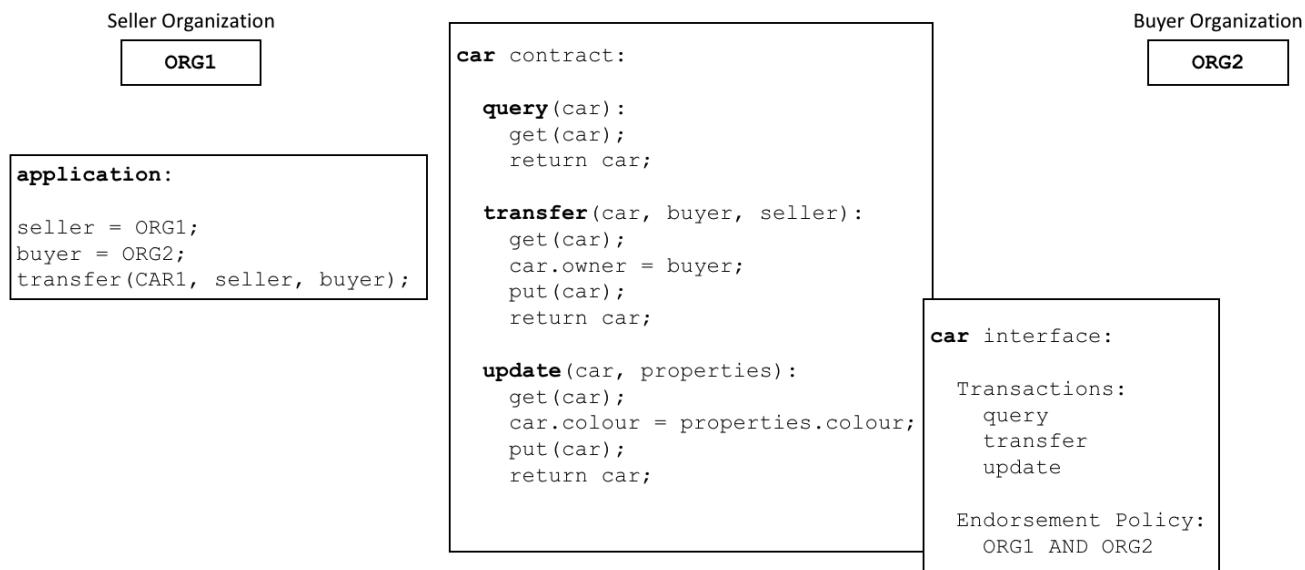
    await ctx.stub.putState(carNumber, Buffer.from(JSON.stringify(car)));
}
```

您可以在“编写第一个应用程序教程”中了解有关 **Fabcar** 智能合约的更多信息。

智能合约可以描述与多组织决策中的数据不变性相关的几乎无限的业务用例。智能合约开发人员的工作是采用可能控制金融价格或交付条件的现有业务流程，并以诸如 JavaScript, GOLANG 或 Java 的编程语言将其表示为智能合约。**精通合同审计的人**越来越多地将数世纪的法律语言转换为编程语言所需的法律和技术技能。您可以在“开发应用程序”主题中了解如何设计和开发智能合约。

5. 背书

与每个链码相关联的是一种签注策略，该策略适用于其中定义的所有智能合约。背书政策非常重要；它指示区块链网络中的哪些组织必须签署由给定的智能合约产生的交易才能宣布该交易 **有效**。



每个智能合约都有与其相关的背书策略。该背书策略可确定哪些组织必须批准智能合约生成的交易，然后才能将其确认为有效交易。

一个示例性的背书策略可以定义参与区块链网络的四个组织中的三个必须在一笔交易被认为**有效**之前对其进行签名。所有**有效**或**无效**交易都将添加到分布式分类帐中，但只有**有效**交易会更新世界状态。

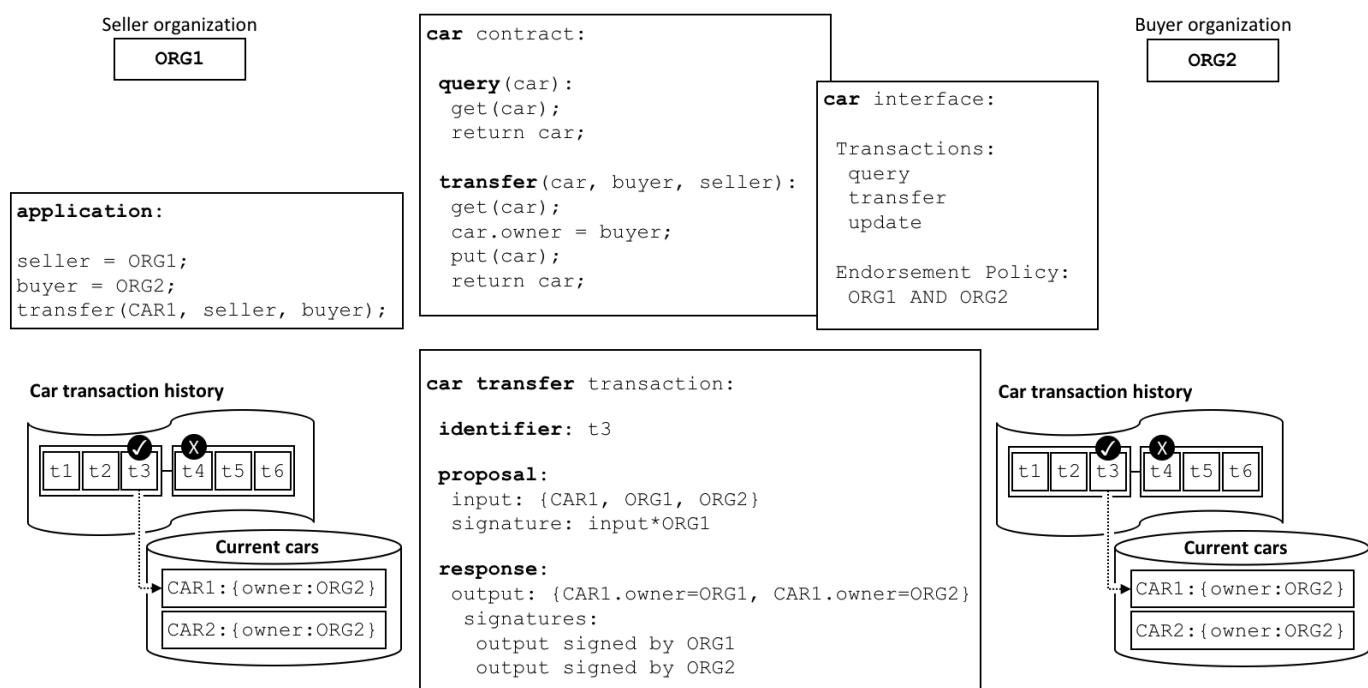
如果背书策略指定必须由多个组织签署交易，则必须由足够多的组织来执行智能合约，以便生成有效的交易。在上面的示例中，与 transfer 汽车的智能合约交易需要双方执行并签名，ORG1 并且 ORG2 该交易有效。

代言政策使 Hyperledger Fabric 与以太坊或比特币等其他区块链有所不同。在这些系统中，网络中的任何节点都可以生成有效的交易。Hyperledger Fabric 更现实地模拟了现实世界；交易必须由网络中的受信任组织验证。例如，政府组织必须签署有效 issueIdentity 交易，或者汽车的 buyer 和 seller 都必须签署 car 转移交易。背书策略旨在使 Hyperledger Fabric 能够更好地对这些类型的实际交互进行建模。

最后，背书策略只是 Hyperledger Fabric 中策略的一个示例。可以定义其他策略以标识谁可以查询或更新分类帐，或从网络中添加或删除参与者。一般而言，尽管不是一成不变的，但政策应由区块链网络中的组织联盟事先达成协议。实际上，策略本身可以定义可以更改策略的规则。尽管是高级主题，但也可以在 Fabric 提供的规则之外定义自定义认可策略规则。

6. 有效交易

当智能合约执行时，它在区块链网络中的一个组织拥有的对等节点上运行。合同采用一组称为交易建议的输入参数，并将其与程序逻辑结合使用以读取和写入分类帐。对世界状态的更改被捕获为 交易建议响应（或仅仅是交易响应），其中包含一个读写集，其中包含已读取的状态以及如果交易有效将要写入的新状态。注意，**执行智能合约时世界状态不会更新！**



所有交易都具有由一组组织签名的标识符，提案和响应。所有交易均记录在区块链上，无论其有效与否，但只有有效交易才有助于世界状态。

检查交易。您可以看到和之间进行汽车换乘的交易。查看交易如何进行输入和输出，以表示所有者从到的更改。请注意如何输入由应用程序的组织签署，并输出由签约双方的赞同政策确定的组织，和。这些签名是通过使用每个参与者的私钥生成的，意味着网络中的任何人都可以验证网络中的所有参与者是否都同意交易细节。`car transfert3ORG1ORG2{CAR1, ORG1, ORG2}{CAR1.owner=ORG1, CAR1.owner=ORG2}ORG1ORG2ORG1ORG1ORG2`

每个对等方在两个阶段中验证分发给网络中所有对等节点的事务。首先，检查交易，以确保有足够的组织根据签注政策签署了该交易。其次，检查以确保世界状态的当前值与由对等节点签名的事务的读取集相匹配；没有中间更新。如果交易通过了这两个测试，则将其标记为有效。所有交易（无论有效还是无效）都添加到区

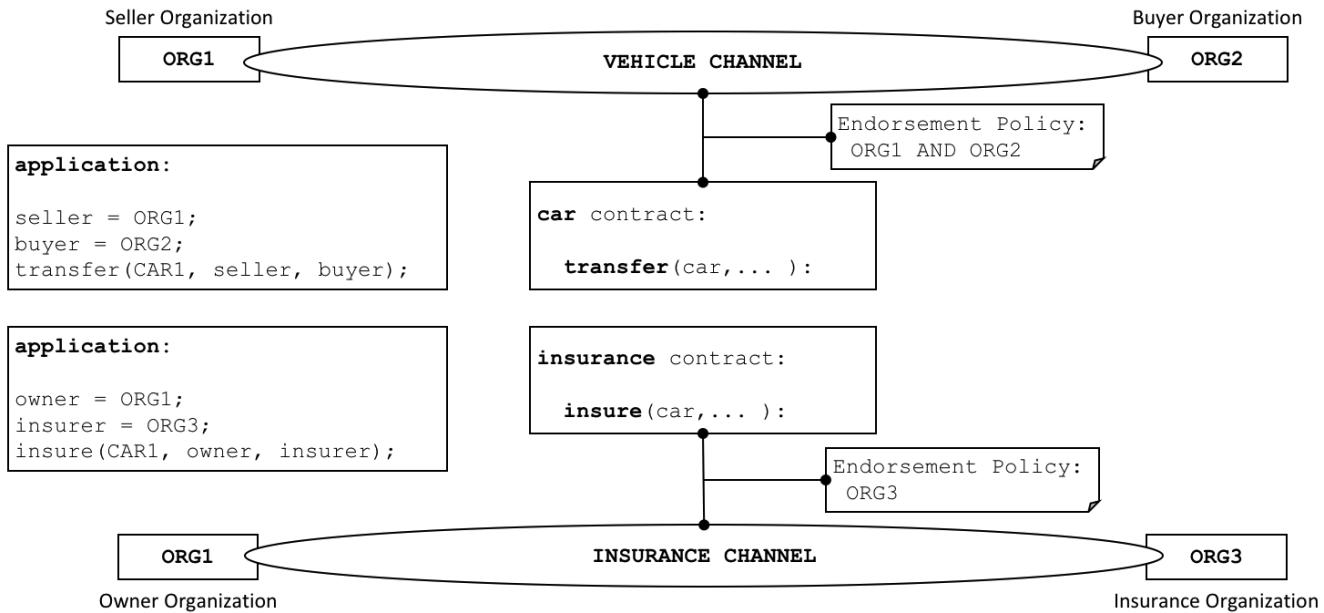
区块链历史中，但是只有**有效**交易才会导致对世界状态的更新。

在我们的示例中，**t3**是有效交易，因此的所有者**CAR1**已更新为**ORG2**。但是，**t4**（未显示）是无效交易，因此，尽管将其记录在分类帐中，但世界状态并未更新，并且**CAR2**仍归**ORG2**。

最后，要了解如何在世界状态下使用智能合约或链码，请阅读[chaincode 命名空间主题](#)。

7. 频道

Hyperledger Fabric 允许组织通过**渠道**同时参与多个独立的区块链网络。通过加入多个渠道，组织可以参与所谓的**网络网络**。通道可有效共享基础架构，同时保持数据和通信的隐私。它们足够独立，可以帮助组织与不同的交易对手分离其工作量，但又足够集成，可以在必要时协调独立的活动。



通道在一组组织之间提供了完全独立的通信机制。将链码定义提交到通道后，链码内的所有智能合约都可用于该通道上的应用程序。

当智能合约代码安装在组织对等方的链码包内时，通道成员只能在通道上定义了链码后才能执行智能合约。该**chaincode 定义**是一个结构，其中包含控制链码操作方式的参数。这些参数包括链码名称，版本和认可策略。每个通道成员通过批准其组织的链码定义来同意链码的参数。当足够多的组织（默认情况下为大多数）批准了相同的链码定义时，可以将该定义提交给渠道。然后，链码内部的智能合约可以由渠道成员执行，但要遵循链码定义中指定的认可策略。认可政策同样适用于同一链代码中定义的所有智能合约。

在上面的示例中，**car** 合同在 **VEHICLE** 渠道上定义，合同在渠道 **insurance** 上定义 **INSURANCE**。的 chaincode 定义 **car** 指定需要双方的认可政策 **ORG1**，并 **ORG2** 签署交易才可以被认为是有效的。 **insurance** 合同的链码定义指定只 **ORG3** 需要背书交易即可。**ORG1** 参与两个网络，该 **VEHICLE** 信道和 **INSURANCE** 网络，并且可以与协调活动 **ORG2** 和 **ORG3** 横跨这两个网络。

链码定义为渠道成员提供了一种方法，使他们在开始使用智能合约在渠道上进行交易之前就链码的管理达成一致。建立在上面的例子中，无论是 **ORG1** 和 **ORG2** 希望认可交易是调用 **car** 合同。由于默认策略要求大多数组织批准链码定义，因此两个组织都需要批准的背书策略 **AND{ORG1,ORG2}**。否则，**ORG1** 和 **ORG2** 将批准不同 chaincode 定义，将无法在 chaincode 定义提交到通道作为一个结果。该过程保证了 **car** 智能合约的交易需要两个组织的批准。

8. 互通

智能合约可以在同一通道内和不同通道之间调用其他智能合约。这样，他们可以读取和写入由于智能合约名称空间而无法访问的世界状态数据。

合同间通信存在局限性，在[chaincode 名称空间](#)主题中已对此进行了全面介绍。

9. 系统链码

链码中定义的智能合约对一组区块链组织之间达成共识的业务流程的域相关规则进行编码。但是，链码还可以定义与领域无关的系统交互相对应的低级程序代码，而这些交互与与业务流程的智能合约无关。

以下是不同类型的系统链码及其相关的缩写：

- [_lifecycle](#) 在所有对等方中运行，并管理对等方上链代码的安装，组织对链代码定义的批准以及链代码定义向渠道的提交。您可以阅读有关如何[_lifecycle](#) 实现 Fabric 链码生命周期[过程的更多信息](#)。
- 生命周期系统链代码（LSCC）管理 Fabric 1.x 版本的链代码生命周期。此版本的生命周期要求链代码在通道上实例化或升级。您可以阅读有关 LSCC 如何实施此[过程的更多信息](#)。如果您将通道应用程序功能设置为 V1_4_x 或更低，您仍然可以使用 LSCC 来管理链码。
- **配置系统链码（CSCC）** 在所有对等方中运行，以处理对通道配置的更改，例如策略更新。您可以在以下[chaincode 主题中](#)了解有关此过程的更多信息。
- **查询系统链码（QSCC）** 在所有对等方中运行，以提供分类账 API，包括区块查询，交易查询等。您可以在交易上下文[主题中](#)了解有关这些分类账 API 的更多信息。
- **背书系统链码（ESCC）** 在背书对等方以加密方式签署交易响应时运行。您可以阅读有关 ESCC 如何实施此[过程的更多信息](#)。
- **验证系统链码（VSCC）** 验证事务，包括检查背书策略和读写集版本控制。您可以阅读有关 LSCC 实现此[过程的更多信息](#)。

底层 Fabric 开发人员和管理员可以修改这些系统链码以供自己使用。但是，系统链代码的开发和管理是一项专门的活动，与智能合约的开发完全分开，通常不需要。对系统链码的更改必须格外小心，因为它们对于 Hyperledger Fabric 网络的正常运行至关重要。例如，如果未正确开发系统链码，则一个对等节点可能会与另一对等节点不同地更新其世界状态或区块链的副本。缺乏共识是**分类帐分叉**的一种形式，这是非常不理想的情况。

分类帐

受众：架构师，应用程序和智能合约开发人员，管理员

一个**总账**是在 Hyperledger 的一个关键的概念；它存储有关业务对象的重要事实信息；对象属性的当前值，以及导致这些当前值的交易历史。

在本主题中，我们将介绍：

- [什么是分类帐？](#)
- [存储有关业务对象的事实](#)
- [区块链分类帐](#)
- [世界状态](#)
- [区块链数据结构](#)
- [区块如何存储在区块链中](#)
- [交易次数](#)
- [世界状态数据库选项](#)
- [该 Fabcar 例如台账](#)
- [分类帐和名称空间](#)
- [分类帐和渠道](#)

1. 什么是分类帐？

分类帐包含业务的当前状态作为交易日志。最早的欧洲和中国分类帐可追溯到大约 1000 年前，而苏美尔人则在 4000 年前就有石分类帐 - 但让我们从一个更新的示例开始吧！

您可能习惯于查看您的银行帐户。对您而言，最重要的是可用余额 - 这是您在当前时刻可以支出的金额。如果要查看余额的产生方式，则可以查看确定余额的交易贷方和借方。这是分类帐的真实示例 - 状态（您的银行余额）和确定分类帐的一组有序交易（贷方和借方）。Hyperledger Fabric 也受这两个方面的影响 - 呈现一组分类帐状态的当前值，并捕获确定这些状态的交易历史。

2. 分类帐，事实和状态

分类帐实际上不存储业务对象，而是存储有关这些对象的**事实**。当我们说“我们将业务对象存储在分类帐中”时，我们真正的意思是正在记录有关对象当前状态的事实以及导致当前状态的交易历史的事实。在一个日益数字化的世界中，感觉就像我们在看一个对象，而不是有关一个对象的事实。对于数字对象，它很可能存在于外部数据存储中。我们存储在分类帐中的事实使我们能够识别其位置以及有关其的其他关键信息。

尽管有关业务对象当前状态的事实可能会发生变化，但是有关它的事实历史是**不可变的**，可以对其进行添加，但不能进行追溯更改。我们将看到如何将区块链视为关于业务对象事实的不变历史，这是一种简单而有效的理解方式。

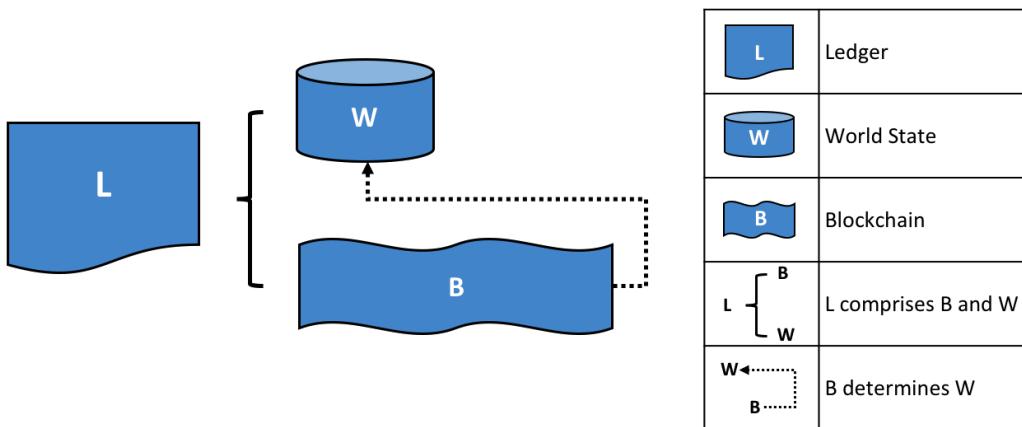
现在让我们仔细看一下 Hyperledger Fabric 分类帐结构！

3. 帐本

在 Hyperledger Fabric 中，分类帐由两个不同但相关的部分组成-世界状态和区块链。这些中的每一个都代表有关一组业务对象的一组事实。

首先，有一个**世界状态**-一个保存一组分类帐状态的**当前值**的数据库。世界状态使程序可以轻松地直接访问状态的当前值，而不必通过遍历整个事务日志来计算状态值。默认情况下，分类帐状态表示为**键值对**，稍后我们将看到 Hyperledger Fabric 如何在这方面提供灵活性。世界状态可以频繁更改，因为可以创建，更新和删除状态。

其次，有一个**区块链**-交易日志，记录了导致当前世界状态的所有更改。交易收集在附加到区块链的区块内部，使您能够了解导致当前世界状态变化的历史。区块链数据结构与世界状态非常不同，因为一旦写入，就无法修改；**这是一成不变的**。



总账 L 由区块链 B 和世界状态 W 组成，其中区块链 B 确定世界状态 W . 我们也可以说世界状态 W 源自区块链 B .

考虑一下 Hyperledger Fabric 网络中存在一个**逻辑**分类帐会很有帮助。实际上，网络维护一个分类帐的多个副本-通过称为**共识**的过程，该副本与其他每个副本保持一致。术语“**分布式分类帐技术**”(DLT) 通常与这种分类帐相关联-这种分类帐在逻辑上是单数形式，但在整个网络中分布有许多一致的副本。

现在让我们更详细地研究世界状态和区块链数据结构。

4. 世界状态

世界状态将业务对象的属性的当前值保留为唯一的分类帐状态。这很有用，因为程序通常需要对象的当前值。遍历整个区块链来计算对象的当前值将很麻烦-您只需直接从世界状态获取即可。

	Ledger world state
{key=CAR1, value=Audi} version=0	A ledger state with key=K . It contains a set of facts expressed as a simple value, V . The state is at version 0.
{key= CAR2, value = {type: BMW, color: red, owner: Jane}} version=0	A ledger state with key=K . It contains a set of facts expressed as a set of key-value pairs {KV} . The state is at version 0.



包含两个州的分类帐世界州。第一种状态是： *key = CAR1* 和 *value = Audi*。第二个状态具有更复杂的值： *key = CAR2* 和 *value = {model: BMW, color = red, owner = Jane}*。两种状态的版本均为 0。

分类帐状态记录有关特定业务对象的一组事实。我们的示例显示了两个汽车 CAR1 和 CAR2 的分类帐状态，每个汽车都有一个键和一个值。应用程序可以调用智能合约，该合约使用简单的分类帐 API 来[获取](#)，[放置](#)和[删除](#)状态。注意状态值如何简单 (Audi...) 或复合 (类型: BMW...)。通常会查询世界状态来检索具有某些属性的对象，例如查找所有红色宝马。

世界状态被实现为数据库。这很有意义，因为数据库为状态的有效存储和检索提供了一组丰富的运算符。稍后我们将看到 Hyperledger Fabric 可以配置为使用不同的世界状态数据库来满足不同类型的状态值和应用程序所需的访问模式的需求，例如在复杂查询中。

应用程序提交捕获世界状态变化的交易，这些交易最终被提交到分类账区块链。Hyperledger Fabric SDK 将应用程序与该[共识](#)机制的细节隔离开来。它们仅调用智能合约，并在交易已包含在区块链中时（无论有效还是无效）得到通知。关键设计要点是，只有由所需的一组[背书组织签名](#)的交易才会导致对世界状态的更新。如果交易没有得到足够的背书人的签名，则不会导致世界状态的改变。您可以阅读有关应用程序如何使用[智能合约](#)的更多信息，以及如何[开发应用程序](#)。

您还将注意到一个状态有一个版本号，在上图中，状态 CAR1 和 CAR2 的起始版本为 0。该版本号供 Hyperledger Fabric 内部使用，并且每次状态更改时都会递增。每当状态更新时都会检查版本，以确保当前状态与认可时的版本匹配。这确保了世界状态正在按预期变化；没有并发更新。

最后，当首次创建分类帐时，世界状态为空。因为任何代表世界状态有效改变的交易都记录在区块链上，这意味着可以随时从区块链重新生成世界状态。这可能非常方便—例如，创建对等体时会自动生成世界状态。此外，如果对等方异常失败，则可以在接受事务之前在对等方重新启动时重新生成世界状态。

5. 区块链

现在，让我们将注意力从世界状态转移到区块链上。世界状态包含与一组业务对象的当前状态有关的一组事实，而区块链是有关这些对象如何到达其当前状态的事实的历史记录。区块链记录了每个分类帐状态的每个先前版本以及更改方式。

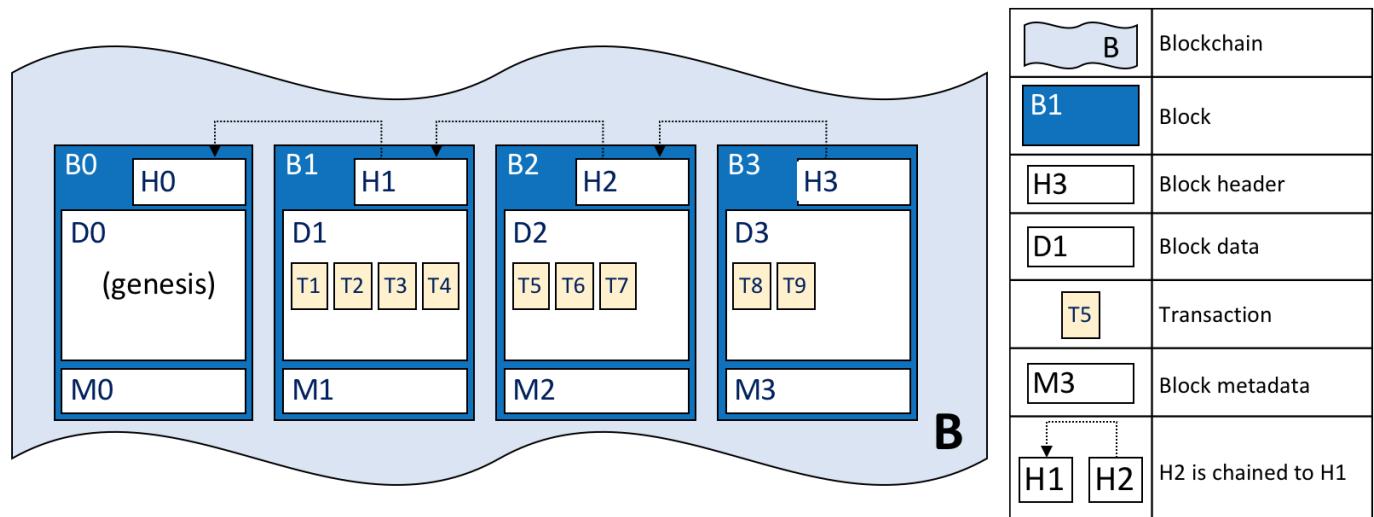
区块链被构造为互连块的顺序日志，其中每个块包含一系列事务，每个事务代表对世界状态的查询或更新。

订购交易的确切机制在[其他地方](#)讨论; 重要的是, 在首次由称为**排序服务**的 Hyperledger Fabric 组件创建块时, 就建立了块排序以及块内的事务排序。

每个区块的标头都包含该区块的交易的哈希值, 以及前一个区块的标头的哈希值。这样, 分类账上的所有交易都被排序并通过密码链接在一起。这种哈希和链接使分类帐数据非常安全。即使托管分类账的一个节点被篡改, 它也无法说服所有其他节点其具有“正确的”区块链, 因为分类账分布在整个独立节点网络中。

与使用数据库的世界状态相反, 区块链始终被实现为文件。这是一个明智的设计选择, 因为区块链数据结构严重偏向极少量的简单操作。追加到区块链的末尾是主要操作, 而查询当前是相对不频繁的操作。

让我们更详细地了解区块链的结构。



包含区块 **B0**, **B1**, **B2**, **B3** 的区块链 **B**. **B0** 是区块链中的第一个区块, 即创世区块。

在上图中, 我们可以看到块 **B2** 的**块数据** **D2** 包含其所有事务: **T5**, **T6**, **T7**。

最重要的是, **B2** 具有**块头** **H2**, 其中包含 **D2** 中所有事务的加密 **哈希**以及 **H1** 的哈希。通过这种方式, 块彼此之间有着千丝万缕的联系, 这就是术语“**区块链**”如此巧妙地捕捉到的!

最后, 如您在图中所看到的, 区块链中的第一个块称为**创世块**。这是分类帐的起点, 尽管它不包含任何用户交易。相反, 它包含一个配置事务, 该事务包含网络通道的初始状态 (未显示)。当我们在文档中讨论区块链网络和[渠道](#)时, 我们将更详细地讨论创世块。

6. 区块

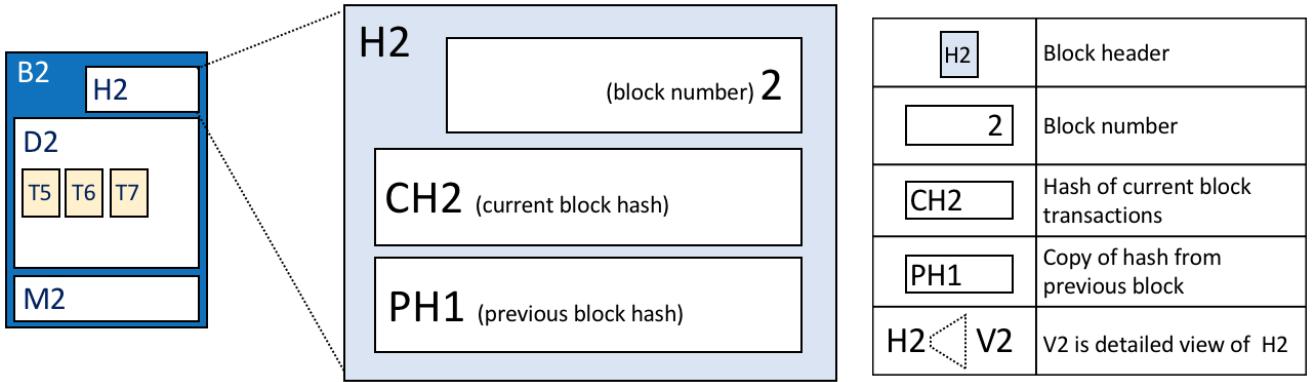
让我们仔细看一下块的结构。它包括三个部分

- **块头**

本节包含三个字段, 在创建块时写入。

- **区块编号**: 一个整数, 从 0 (创世区块) 开始, 并随附加到区块链的每个新区块增加 1。
- **当前块哈希**: 当前块中包含的所有事务的哈希。
- **上一个块标题的哈希值**: 上一个块标题的哈希值。

这些字段是通过对块数据进行密码哈希处理而在内部派生的。他们确保每个街区都与其邻居密不可分, 从而导致总账不变。



- 块标题详细信息。块 B_2 的头 H_2 由块号 2, 当前块数据 D_2 的哈希 CH_2 和先前块头 H_1 的哈希组成。
- 块数据**

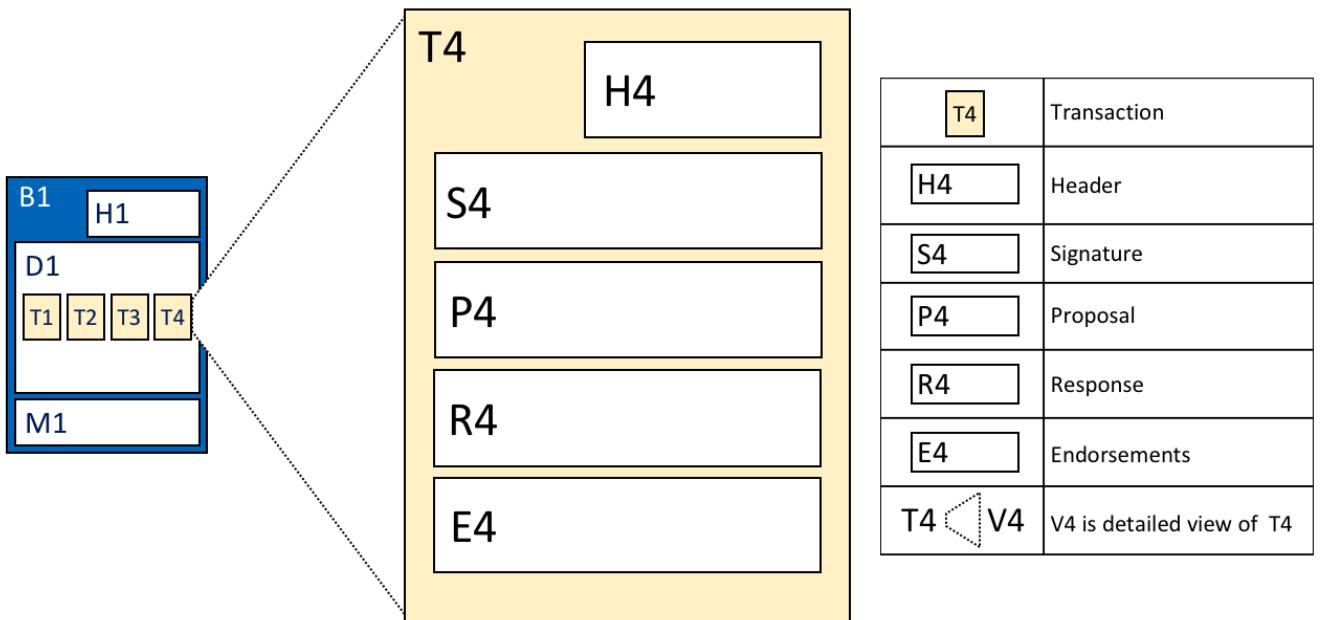
本节包含按顺序排列的事务列表。在排序服务创建块时写入。这些事务具有丰富但直接的结构，我们将在本主题的后面部分进行介绍。

• 区块元数据

此部分包含写入块的时间，以及块编写器的证书，公共密钥和签名。随后，块提交者还为每个事务添加有效/无效指示符，尽管此信息不包括在哈希中，因为创建块时会创建该信息。

7. 交易次数

如我们所见，事务捕获了世界状态的变化。让我们来看看详细的 **BLOCKDATA** 包含在一个块中的交易结构。



易明细。块 B_1 的块数据 D_1 中的事务 T_4 由事务头 H_4 , 事务签名 S_4 , 事务建议 P_4 , 事务响应 R_4 和背书列表 E_4 组成。

在上面的示例中，我们可以看到以下字段：

- **标头**

由 H4 举例说明的本节捕获有关交易的一些基本元数据，例如，相关链码的名称及其版本。

- **签名**

由 S4 说明的该部分包含由客户端应用程序创建的加密签名。此字段用于检查交易明细是否未被篡改，因为它需要应用程序的私钥来生成它。

- **提案**

用 P4 表示的该字段对应用程序提供给创建拟议分类账更新的智能合约的输入参数进行编码。当智能合约运行时，此提议提供一组输入参数，这些输入参数与当前的世界状态一起确定新的世界状态。

- **响应**

用 R4 表示的此部分将世界状态的前后值捕获为**读写集** (RW-set)。它是智能合约的输出，如果交易成功通过验证，它将应用于分类账以更新世界状态。

- **背书**

如图 E4 所示，这是来自每个所需组织的足以满足背书策略的已签名交易响应的列表。您会注意到，虽然事务中仅包含一个事务响应，但有多个背书。这是因为每个背书都有效地编码了其组织的特定交易响应-意味着不需要包含任何与足够背书不匹配的交易响应，因为它将被拒绝为无效交易，并且不会更新世界状态。

总结了交易的主要领域-还有其他领域，但是您必须了解这些必不可少的领域，才能对分类账数据结构有深入的了解。

8. 世界状态数据库选项

世界状态从物理上实现为数据库，以提供简单有效的存储和分类账状态检索。如我们所见，分类帐状态可以具有简单值或复合值，并且为了适应这种情况，世界状态数据库的实现可以变化，从而可以有效地实现这些值。世界状态数据库的选项当前包括 LevelDB 和 CouchDB。

LevelDB 是默认值，当分类帐状态是简单的键/值对时尤其适用。LevelDB 数据库与对等节点位于同一位置-嵌入在同一操作系统进程中。

当分类帐状态被构造为 JSON 文档时，CouchDB 是一个特别合适的选择，因为 CouchDB 支持丰富的查询和更新业务交易中经常发现的更丰富的数据类型。在实现方面，CouchDB 在单独的操作系统进程中运行，但是对等节点和 CouchDB 实例之间仍然存在 1: 1 的关系。所有这些对于智能合约都是看不见的。有关 [CouchDB 的](#) 更多信息，请参见 [CouchDB 作为 StateDatabase](#)。

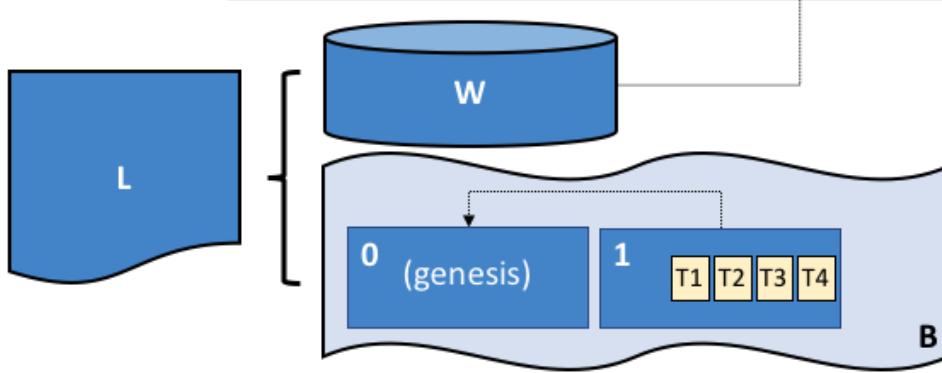
在 LevelDB 和 CouchDB 中，我们看到了 Hyperledger Fabric 的一个重要方面-它是可插入的。世界状态数据库可以是关系数据存储，图形存储或时间数据库。这为可以有效访问的分类帐状态类型提供了极大的灵活性，从而使 Hyperledger Fabric 可以解决许多不同类型的问题。

9. 分类帐示例：fabcar

当我们在分类账上结束本主题时，让我们看一下样本分类账。如果您运行了 [fabcar 示例应用程序](#)，那么您已经创建了此分类帐。

fabcar 示例应用程序创建了一组 10 辆汽车，每辆汽车具有唯一的标识；不同的颜色，品牌，型号和所有者。这是创建前四辆车后的分类帐外观。

key=CAR3, value={color: yellow, make: Volkswagen, model: Passat, owner: Max}	version=0
key=CAR2, value={color: green, make: Hyundai, model: Tucson, owner: Jin Soo}	version=0
key=CAR1, value={color: red, make: Ford, model: Mustang, owner: Brad}	version=0
key=CAR0, value={color: blue, make: Toyota, model: Prius, owner: Tomoko}	version=0



分类帐 **L** 包含一个世界状态 **W** 和一个区块链 **B**。 **W** 包含四个具有键的状态: **CAR0, CAR1, CAR2** 和 **CAR3**。 **B** 包含两个块 **0** 和 **1**。 块 **1** 包含四个事务: **T1, T2, T3, T4**。

我们可以看到世界状态包含对应于 **CAR0, CAR1, CAR2** 和 **CAR3** 的状态。 **CAR0** 的值表示它是 **Tomomo** 目前拥有的蓝色 **Toyota Prius**，我们可以看到其他汽车的相似状态和值。此外，我们可以看到所有汽车状态均为版本号 **0**，表明这是它们的起始版本号-自创建以来尚未对其进行更新。

我们还可以看到，区块链包含两个块。块 **0** 是创世区块，尽管它不包含任何与汽车相关的交易。但是，块 **1** 包含事务 **T1, T2, T3, T4**，这些事务对应于在世界状态下为 **CAR0** 到 **CAR3** 创建初始状态的事务。我们可以看到块 **1** 链接到块 **0**。

我们没有显示块或事务中的其他字段，特别是标头和哈希。如果您对这些内容的精确细节感兴趣，可以在文档的其他地方找到专用的参考主题。它为您提供了整个块的完整工作示例，其中包含详细的交易细节-但到目前为止，您已经对 Hyperledger Fabric 分类帐有了扎实的概念性理解。做得好！

10. 命名空间

即使我们已经将分类帐呈现为一个单一的世界状态和单个区块链，但这还是有点过分简化了。实际上，每个链码都有自己的世界状态，该世界状态与所有其他链码是分开的。世界状态位于名称空间中，因此只有相同链代码内的智能合约才能访问给定的名称空间。

区块链未命名空间。它包含来自许多不同的智能合约名称空间的事务。您可以在本[主题中](#)阅读有关链码名称空间的更多信息。

现在让我们看一下如何在 Hyperledger Fabric 通道中应用名称空间的概念。

11. 频道

在 Hyperledger Fabric 中，每个[通道](#)都有一个完全独立的分类帐。这意味着完全独立的区块链，以及完全独立的世界状态，包括名称空间。应用程序和智能合约可以在渠道之间进行通信，以便可以在它们之间访问分类帐信息。

在本[主题中](#)，您可以阅读有关分类帐如何与渠道一起使用的更多信息。

12. 更多信息

请参阅[事务流](#), [读写集语义](#)和[CouchDB 作为 StateDatabase 主题](#), 以更深入地了解事务流, 并发控制和世界状态数据库。

排序服务

受众：建筑师，排序服务管理员，渠道创建者

本主题从概念上介绍了排序的概念，排序者如何与对等方交互，他们在交易流中扮演的角色以及排序服务的当前可用实现的概述，尤其着重于 Raft 排序服务。实施。

1. 排序什么？

不允许以太坊和比特币等许多分布式区块链，这意味着任何节点都可以参与共识过程，在此过程中，交易被排序并捆绑成块。因此，这些系统依赖于**概率**共识算法，该算法最终可以确保分类帐的一致性达到很高的概率，但是仍然容易受到发散分类帐（也称为分类“分叉”）的影响，因为网络中的不同参与者具有不同的分类帐。对接受的交易顺序的不同看法。

Hyperledger Fabric 的工作原理有所不同。它具有一个称为“**排序者**”的节点（也称为“排序节点”）来执行此交易订购，该节点与其他**排序节点**一起构成**排序服务**。由于 Fabric 的设计依赖于**确定性**共识算法，因此可以保证对等方验证的任何块都是最终的和正确的。账本管理者无法像在许多其他分布式和无许可的区块链网络中那样做。

除了提高最终性之外，将链码执行的认可（在对等点发生）与订购分开可以为 Fabric 提供性能和可伸缩性方面的优势，消除了在同一节点执行和订购时可能出现的瓶颈。

2. 排序者节点和渠道配置

除其**排序角色**外，**排序者**还维护被允许创建渠道的组织的列表。该组织列表称为“**联盟**”，该列表本身保留在“**排序节点系统渠道**”（也称为“**排序系统渠道**”）的配置中。默认情况下，该列表及其所处的频道只能由排序者管理员编辑。请注意，排序服务可能会保留其中几个列表，这使该联盟成为 Fabric 多租户的工具。

排序者还对通道实施基本的访问控制，从而限制了谁可以对其进行读写数据以及谁可以对其进行配置。请记住，有权修改通道中的配置元素的人员必须受相关管理员在创建联盟或通道时设置的策略的约束。配置事务由排序节点处理，因为排序者需要知道当前策略集才能执行其基本形式的访问控制。在这种情况下，排序者将处理配置更新，以确保请求者具有适当的管理权限。如果是这样，则排序节点针对现有配置验证更新请求，生成新的配置事务，并将其打包到一个块中，该块中继到通道上的所有对等方。

3. 排序者节点和身份

与区块链网络交互的所有事物（包括对等方，应用程序，管理员和排序节点）都从其数字证书和其成员资格服务提供商（MSP）定义中获取其组织身份。

有关身份和 MSP 的更多信息，请查看有关[身份](#)和[成员资格](#)的文档。

就像对等体一样，排序节点属于组织。与对等节点一样，每个组织都应使用单独的证书颁发机构（CA）。此 CA 是否将充当根 CA，还是您选择部署根 CA，然后选择与该根 CA 关联的中间 CA，取决于您。

4. 排序者和交易流程

第一阶段：提案

从关于 [Peers](#) 的主题中我们已经看到，它们构成了区块链网络的基础，托管分类帐，应用程序可以通过智能合约查询和更新分类帐。

具体来说，想要更新分类帐的应用程序涉及三个阶段的过程，以确保区块链网络中的所有对等方都保持其分类帐彼此一致。

在第一阶段，客户端应用程序将交易建议发送给对等方的子集，该对等方将调用智能合约以产生建议的分类帐更新，然后对结果进行背书。背书的对等方目前不将建议的更新应用于其账本副本。相反，背书的对等方将投标响应返回到客户端应用程序。背书的交易建议将最终在第二阶段按顺序排列，然后分发给所有对等方以进行最终验证并在第三阶段进行提交。

要深入了解第一阶段，请参考 [Peers](#) 主题。

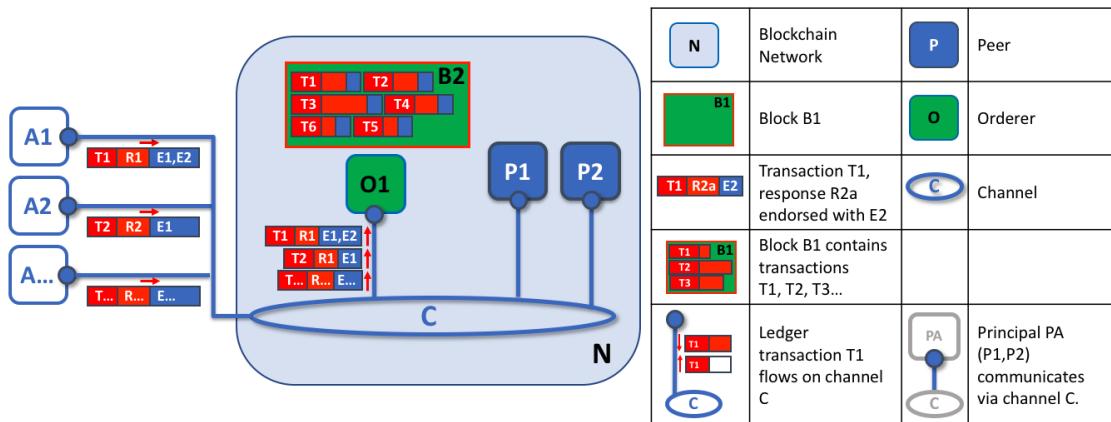
第二阶段：将交易排序和打包成块

在交易的第一阶段完成之后，客户端应用程序已收到来自一组对等方的认可交易建议响应。现在该进行交易的第二阶段了。

在此阶段，应用程序客户端将包含认可交易建议响应的交易提交到排序服务节点。排序服务创建交易块，最终将这些交易块分发给通道上的所有对等方，以进行最终验证并在第三阶段进行提交。

排序服务节点同时接收来自许多不同应用程序客户端的事务。这些排序服务节点一起工作以共同形成排序服务。它的工作是将提交的事务的批处理按定义明确的顺序排列，然后打包成块。这些区块将成为区块链的区块！

记录，在一个块的数量取决于与所期望的大小和最大经过的持续时间为一个块信道的配置参数（[BatchSize](#) 和 [BatchTimeout](#) 参数，是精确的）。然后将这些块保存到排序节点的分类帐中，并分发给已加入该渠道的所有对等方。如果对等节点此时恰好处于关闭状态，或者稍后加入了该通道，则在重新连接到排序服务节点后，或者与另一对等节点闲聊，它将接收到块。在第三阶段，我们将看到对等方如何处理此块。



排序节点的第一个角色是打包建议的分类帐更新。在此示例中，应用程序 A1 将由 E1 和 E2 认可的交易 T1 发送给排序节点 O1。并行地，应用程序 A2 将由 E1 认可的事务 T2 发送到排序节点 O1。O1 将来自应用程序 A1 的事务 T1 和来自应用程序 A2 的事务 T2 以及来自网络中其他应用程序的其他事务打包到块 B2 中。我们可以看到，在 B2 中，交易顺序为 T1, T2, T3,

T4, T6, T5 – 可能不是这些交易到达排序节点的顺序！（此示例显示了非常简化的排序服务配置，其中只有一个排序节点。）

值得注意的是，一个区块中交易的顺序不一定与排序服务所接收的订单相同，因为可能有多个排序服务节点大约在同一时间接收交易。重要的是排序服务将交易置于严格的顺序中，对等方将在验证和提交交易时使用此顺序。

区块内交易的这种严格排序使 Hyperledger Fabric 与其他区块链略有不同，在其他区块链中，同一笔交易可以打包成多个不同的区块，竞争形成一个链。在 Hyperledger Fabric 中，排序服务生成的块是 **final**。一旦将交易写入一个区块，就可以确保其在分类账中的位置。如前所述，Hyperledger Fabric 的终结性意味着没有 **分类账分叉** - 验证的交易将永远不会被还原或丢弃。

我们还可以看到，尽管对等节点执行智能合约并处理交易，但排序节点绝对不会这样做。到达排序节点的每个授权交易都被机械地打包在一个块中-排序节点不对交易的内容做出判断（如前所述，通道配置交易除外）。

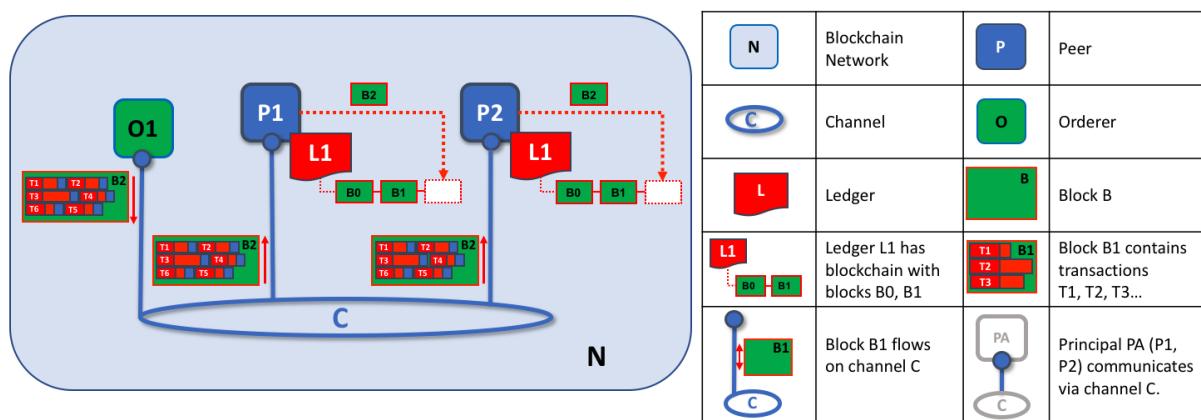
在第二阶段结束时，我们看到排序节点负责简单但至关重要的过程，这些过程包括收集建议的交易更新，进行订购并将它们打包成块以便分发。

第三阶段：验证和提交

交易工作流程的第三阶段涉及从排序节点到对等方的块的分发和后续验证，在这里可以将它们提交到分类账。

阶段 3 从排序节点将块分配给与其连接的所有对等点开始。还值得注意的是，并非每个对等点都需要连接到排序节点，对等点可以使用 **八卦** 协议将块级联到其他对等点。

每个对等方将独立地但以确定性的方式验证分布式块，以确保分类账保持一致。具体来说，渠道中的每个对等方都将验证区块中的每个交易，以确保其已被所需组织的对等方认可，其背书相匹配，并且未被其他最近提交的交易（可能已经在交易中进行的交易）使之无效。最初批准交易时的航班。无效的交易仍保留在排序节点创建的不可变块中，但对等方将其标记为无效，并且不会更新分类帐的状态。



排序节点的第二个作用是将块分配给同级。在该示例中，定购者 O1 将块 B2 分配给对等体 P1 和对等体 P2。对等体 P1 处理块 B2，从而将新块添加到 P1 上的分类帐 L1。并行地，对等体 P2 处理块 B2，导致将新块添加到 P2 上的分类帐 L1。一旦该过程完成，分类帐 L1 就已在对等端 P1 和 P2 上进行了一致更新，并且每个分类帐可以通知连接的应用程序交易已处理。

总而言之，在第三阶段中，排序服务生成的模块始终应用于分类帐。严格按顺序将交易划分为块，每个对

等方都可以验证交易更新是否在整个区块链网络中得到一致应用。

要深入了解第 3 阶段, 请返回 [Peers](#) 主题。

5. 排序服务实施

尽管当前可用的每个排序服务都以相同的方式处理事务和配置更新, 但是仍然存在几种不同的实现方式, 用于在排序服务节点之间对严格的事务订购达成共识。

有关如何站起排序节点的信息 (无论使用哪种实现节点), 请查阅[我们有关站起排序节点的文档](#)。

- **Raft**

新的 1.4.1 的, Raft 是根据实现方式的碰撞容错 (CFT) 排序服务 [RAFT 协议](#) 在 [etcd](#)。RAFT 遵循

“领导者和跟随者” 模型, 其中 (每个通道) 选举领导者节点, 并由跟随者复制其决策。与基于 Kafka 的排序服务相比, Raft 排序服务应该更易于设置和管理, 并且其设计允许不同的组织将节点贡献给分布式排序服务。

- **卡夫卡**

类似于基于 Raft 的订购, Apache Kafka 是一种 CFT 实现, 它使用 “领导者和跟随者” 节点配置。

Kafka 利用 ZooKeeper 集合进行管理。从 Fabric v1.0 开始可以使用基于 Kafka 的排序服务, 但是许多用户可能会发现管理 Kafka 群集的额外管理开销令人生畏或不受欢迎。

- **SOLO**

排序服务的 Solo 实现仅用于测试, 并且仅包含一个排序节点。它已被弃用, 在将来的版本中可能会完全删除。Solo 的现有用户应移至单节点 Raft 网络以实现等效功能。

6. RAFT

有关如何配置 Raft 排序服务的信息, 请查看 [有关配置 Raft 排序服务的文档](#)。

生产网络的排序服务选择, 已建立的 Raft 协议的 Fabric 实现使用“领导者和跟随者”模型, 在该模型中, 领导者是在渠道中的排序节点之间动态选举的 (这种节点集合称为 (“同意集”), 并且该领导者将消息复制到跟随者节点。因为只要剩余大多数排序节点 (称为“仲裁”), 系统就可以承受包括引导节点在内的节点的丢失, 因此 Raft 被称为“故障容错” (CFT)。换句话说, 如果一个通道中有三个节点, 则它可以承受一个节点的丢失 (剩下两个)。如果通道中有五个节点, 则可能会丢失两个节点 (剩下三个剩余节点)。

从它们提供给网络或渠道的服务的角度来看, Raft 与现有的基于 Kafka 的排序服务 (我们将在后面讨论) 相似。它们都是使用领导者和跟随者设计的 CFT 排序服务。如果您是应用程序开发人员, 智能合约开发人员或对等管理员, 则不会注意到基于 Raft 和 Kafka 的排序服务之间的功能差异。但是, 有一些主要差异值得考虑, 尤其是如果您打算管理排序服务:

- RAFT 更容易安装。尽管 Kafka 拥有许多崇拜者, 但即使是那些崇拜者, 他们也 (通常) 会承认部署 Kafka 集群及其 ZooKeeper 集成可能很棘手, 这需要在 Kafka 基础架构和设置方面具有高水平的专业知识。此外, 与 Raft 相比, Kafka 需要管理的组件更多, 这意味着出错的地方更多。Kafka 有其自己的版本, 必须与您的排序节点协调。**使用 Raft, 一切都将嵌入到您的排序节点中。**
- Kafka 和 Zookeeper 并非旨在跨大型网络运行。它们被设计为 CFT, 但应在紧密的主机中运行。这就是说, 实际上, 您需要一个组织来运行 Kafka 集群。鉴于此, 在使用 Kafka (Fabric 支持) 的情况下, 让排序节点由不同的组织运行不会带来很多分散性, 因为节点将全部进入同一个由单个组织控制的 Kafka 集群。借助 Raft, 每个组织都可以拥有自己的排序节点, 参与排序服务, 从而导致系统更加分散。

- RAFT 本身受支持。尽管基于 Kafka 的排序服务当前与 Fabric 兼容，但要求用户获取必需的图像并了解如何自行使用 Kafka 和 ZooKeeper。同样，对 Kafka 相关问题的支持是通过 Kafka 的开源开发人员 [Apache](#) 而非 Hyperledger Fabric 进行的。另一方面，已经开发了 Fabric Raft 实施，并将在 Fabric 开发人员社区及其支持机构中提供支持。
- Kafka 使用服务器池（称为“Kafka 经纪人”），而排序节点组织的管理员指定他们要在特定频道上使用多少个节点，Raft 允许用户指定将哪个排序节点部署到哪个频道。这样，对等组织可以确保，如果他们还拥有一个排序节点，则将该节点作为该渠道的排序服务的一部分，而不是信任并依靠中央管理员来管理 Kafka 节点。
- RAFT 是 Fabric 开发拜占庭式容错（BFT）排序服务的第一步。就像我们将看到的那样，Raft 开发中的某些决定是由此驱动的。如果您对 BFT 感兴趣，则学习如何使用 Raft 应该可以简化过渡过程。

注意：与 Solo 和 Kafka 相似，RAFT 排序服务在收到回执后会丢失交易。例如，如果领导者大约在追随者提供收据确认的同时崩溃。因此，应用程序客户端无论如何都应在对等方上侦听事务提交事件（以检查事务的有效性），但应格外小心，以确保客户端也能容忍超时，该超时不会在配置的时间范围内落实事务。取决于应用程序，可能希望在这种超时情况下重新提交交易或收集一组新的认可。

7. RAFT 概念

尽管 Raft 提供了许多与 Kafka 相同的功能-尽管采用了更简单易用的包装-但在 Kafka 的掩盖下其功能却大不相同，并向 Fabric 引入了许多新概念或对现有概念的扭曲。

日志条目。 Raft 排序服务中的主要工作单元是“日志条目”，此类条目的完整顺序称为“日志”。如果大多数成员（换言之为法定人数）同意条目及其顺序，则我们认为日志是一致的，从而使复制了各种排序节点的日志。

同意集。 排序节点积极参与给定通道的共识机制，并接收该通道的复制日志。这可以是所有可用节点（在单个群集中或在组成系统通道的多个群集中），也可以是那些节点的子集。

有限状态机（FSM）。 Raft 中的每个排序节点都有一个 FSM，它们共同用于确保各个排序节点中日志的顺序是确定性的（以相同顺序编写）。

法定人数。 描述需要确认提案以允许订购交易的最小同意者数量。对于每个同意集，这是 **大多数** 节点。在具有五个节点的群集中，三个群集必须可用。如果由于某种原因无法达到法定数量的节点，则排序服务集群将无法用于通道上的读取和写入操作，并且无法提交任何新日志。

领队。 这不是一个新概念-正如我们所说的，Kafka 还使用了领导者-但至关重要的是要了解，在任何给定时间，渠道的同意者集会选举一个节点作为领导者（我们将描述这种情况是如何发生的）RAFT 稍后）。负责人负责摄取新的日志条目，将它们复制到关注者排序节点，并管理何时将条目视为已提交。这不是订购器的特殊类型。这只是排序节点可能在某些时候扮演的角色，而在其他情况下（视情况而定）则没有。

追随者。 同样，这不是一个新概念，但了解追随者的关键是追随者从领导者那里接收日志并确定性地复制它们，以确保日志保持一致。正如我们将在领导者选举一节中看到的那样，关注者还会从领导者那里收到“心跳”消息。如果领导者在可配置的时间内停止发送这些消息，则追随者将发起领导者选举，其中一个将被选举为新领导者。

8. 交易流程中的 RAFT

每个通道都在 Raft 协议的**单独**实例上运行，该协议允许每个实例选举不同的领导者。在群集由不同组织

控制的排序节点组成的用例中，此配置还允许进一步分散服务。尽管所有 Raft 节点都必须是系统通道的一部分，但不一定必须是所有应用程序通道的一部分。频道创建者（和频道管理员）可以选择可用排序节点的子集，并根据需要添加或移除排序节点（只要一次仅添加或移除一个节点）。

尽管此配置以冗余心跳消息和 goroutine 的形式创建了更多开销，但为 BFT 奠定了必要的基础。

在 Raft 中，交易（以提议或配置更新的形式）由接收交易的排序节点自动路由到该通道的当前负责人。这意味着对等方和应用程序不需要在任何特定时间知道谁是领导者节点。仅排序节点需要知道。

完成排序节点验证检查后，将按照我们的交易流程第二阶段中的说明对交易进行订购，打包，成块同意和分发。

9. 架构师笔记

RAFT 选举中领导人选举的工作方式

尽管选举领导者的过程发生在排序节点的内部流程中，但值得注意的是该流程是如何进行的。

RAFT 节点始终处于以下三种状态之一：跟随者，候选者或领导者。所有节点最初都是作为 **跟随者** 开始的。在这种状态下，他们可以接受来自领导者的日志条目（如果已当选），或为领导者投票。如果在设定的时间段内（例如，五秒钟）未接收到日志条目或心跳，则节点会自动升级为 **候选** 状态。在候选状态下，节点向其他节点请求投票。如果候选人获得法定人数的选票，则将其晋升为 **领导人**。领导者必须接受新的日志条目并将其复制到关注者。

要直观了解领导者选举过程的工作方式，请查看 [“数据的秘密生活”](#)。

快照

如果排序节点出现故障，它如何获取重新启动时丢失的日志？

尽管可以无限期地保留所有日志，但是为了节省磁盘空间，Raft 使用了一个称为“快照”的过程，用户可以在其中定义将在日志中保留多少字节的数据。此数据量将符合一定数量的块（取决于块中的数据量。请注意，快照中仅存储完整的块）。

例如，假设滞后的副本 R1 刚刚重新连接到网络。它的最新块是 100。领导者 L 位于区块 196，并配置为以表示 20 个区块的数据量进行快照。R1 因此将接收块 180 从 L 再做出 Deliver 用于块的请求 101 到 180。块 180 到 196 将被复制到 R1 通过正常 Raft 协议。

卡夫卡

Fabric 支持的另一个崩溃容错排序服务是对 Kafka 分布式流平台的改编，可以用作排序节点的集群。您可以在 [Apache Kafka 网站上](#) 了解有关 Kafka 的更多信息，但从更高层次上讲，Kafka 使用与 Raft 相同的概念“领导者和跟随者”配置，其中从领导者复制事务（Kafka 称为“消息”）。节点到跟随者节点。如果领导节点发生故障，则跟随者之一将成为领导者，订单可以继续进行，从而确保了容错能力，就像 Raft 一样。

Kafka 集群的管理，包括任务的协调，集群成员，访问控制和控制器选举等，均由 ZooKeeper 集成及其相关 API 进行处理。

众所周知，Kafka 集群和 ZooKeeper 集成很难设置，因此我们的文档假定您对 Kafka 和 ZooKeeper 有一定的了解。如果您决定在没有专业知识的情况下使用 Kafka，则在尝试使用基于 Kafka 的排序服务之前，至少应完成《[Kafka 快速入门](#)》指南的前六个步骤。您也可以查阅 [此样本配置文件](#)，以简要了解 Kafka 和 ZooKeeper

的合理默认值。

要了解如何启动基于 Kafka 的排序服务，请查看[有关 Kafka 的文档](#)。

私人数据

1. 什么是私人数据?

如果某个渠道上的一组组织需要使该数据与该渠道上的其他组织保持私有，则可以选择创建一个仅包含需要访问数据的组织的新渠道。但是，在每种情况下创建单独的渠道都会产生额外的管理开销(维护链码版本，策略，MSP等)，并且不允许您希望所有渠道参与者在保留一部分资产的情况下查看交易的用例。私人数据。

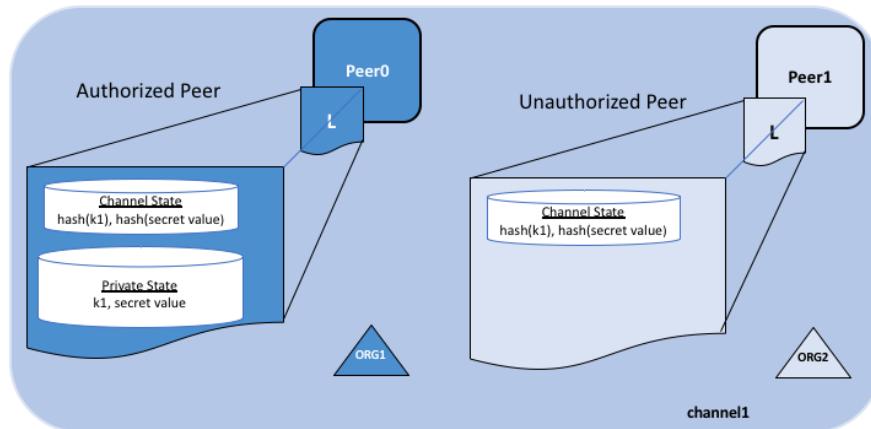
因此，从v1.2开始，Fabric提供了创建**私有数据集合的功能**，该功能使渠道上已定义的组织子集能够认可，提交或查询私有数据，而无需创建单独的渠道。

2. 什么是私人数据收集?

集合是两个元素的组合：

1. **实际的私人数据**通过八卦协议对等发送到仅授权查看该数据的组织。此数据存储在授权组织的对等方的私有状态数据库中（有时称为“边”数据库或“SideDB”），可以从这些授权对等方上的链码进行访问。此处不涉及排序服务，并且看不到私有数据。请注意，由于八卦会在授权组织之间分配对等私有数据，因此需要引导通道上的锚点对等点，并在每个对等点上配置CORE_PEER_GOSSIP_EXTERNALENDPOINT，以引导跨组织通信。
2. **该数据的哈希值**，已被认可，排序并写入通道中每个对等方的分类账中。散列用作事务处理的证据，用于状态验证，并可用于审计目的。

下图说明了被授权拥有私有数据的对等实体的分类账内容，而没有私有数据的对等账本内容。



如果收集成员发生争议或想要将资产转让给第三方，则收集成员可以决定与其他方共享私人数据。然后，第三方可以计算私有数据的哈希值，并查看其是否与通道分类帐上的状态匹配，从而证明该状态在某个时间点存在于集合成员之间。

何时在一个通道内使用一个集合与一个单独的通道上的集合

- 使用**渠道**，当整个交易（和分类账）必须一组是渠道成员的组织内予以保密。
- 当必须在一组组织之间共享事务（和分类帐）时，但是当这些组织的子集应有权访问事务中的某些（或全部）数据时，请使用**集合**。此外，由于私有数据是通过点对点而不是通过块进行分发的，因此当必须对交易数据保密时，请使用私有数据集合，以免排序服务节点。

3. 用例解释集合

考虑一个在渠道上交易产品的五个组织：

- **一个农民**在国外出售商品
- **分销商**将货物移至国外
- **托运人**在各方之间移动货物
- **批发商**从分销商处购买商品
- **零售商**从托运人和批发商那里购买商品

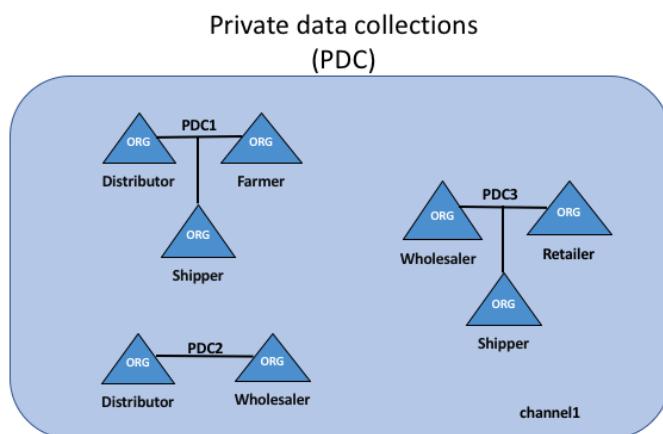
该**分销商**可能希望使与私人事务 **农民**和**托运人**保持在机密的交易的条款**批发商**和**零售商**（以免暴露他们正在充电的标记）。

该**经销商**还可能希望与一个独立的私人数据关系的**批发商**因为只要他们以较低的价格比它的 **零售商**。

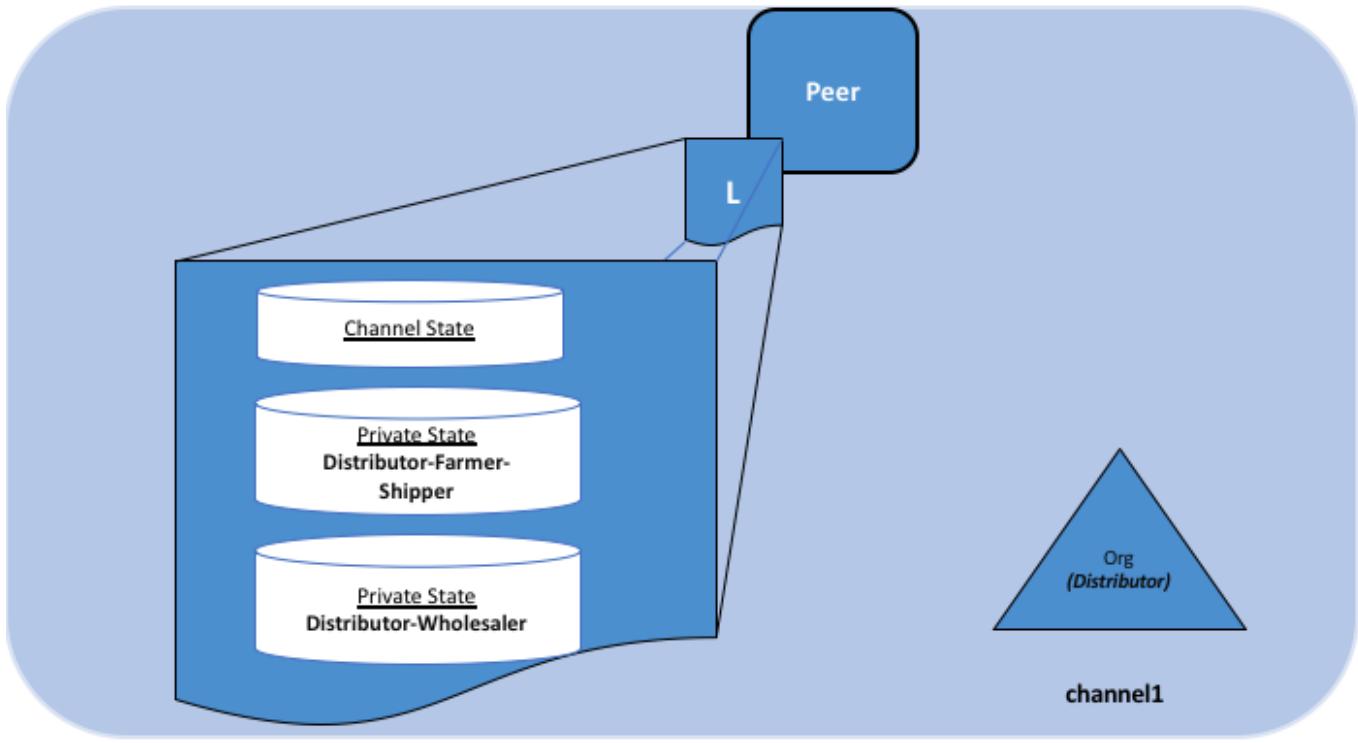
该**批发商**还可能希望与私有数据关系 **零售商**和**托运人**。

可以为多个关系之间定义共享多个私有数据集合（**PDC**），而不是为这些关系中的每个定义许多小渠道。

1. PDC1: **分销商**, **农民**和**托运人**
2. PDC2: **分销商**和**批发商**
3. PDC3: **批发商**, **零售商**和**托运人**



使用此示例，**分销商**拥有的对等方将在其分类账内拥有多个私有数据库，其中包括来自**分销商**，**农民**和**托运人**关系以及**分销商**和**批发商**关系的私有数据。因为这些数据库与保存通道分类帐的数据库是分开的，所以有时将私有数据称为“SideDB”。



4. 带有私人数据的交易流

当在链码中引用私有数据集合时，事务流程略有不同，以便在提议，认可并提交到分类帐时保护私有数据的机密性。

有关不使用私有数据的交易流的详细信息，请参阅我们的[交易流](#)文档。

1. 客户端应用程序提交提案请求以调用链码功能（读取或写入私有数据）给背书的对等方，这些对等方是集合的授权组织的一部分。私有数据或用于以链码生成私有数据的数据在投标 `transient` 字段中发送。
2. 认可对等方模拟事务并将私有数据存储在对等方本地的临时存储中。他们根据收集策略通过 `gossip` 将私有数据分发给授权的对等方。`transient data store`
3. 认可对等方将提案响应发送回客户端。投标响应包括认可的读/写集，其中包括公共数据，以及任何私有数据键和值的哈希。没有私有数据发送回客户端。有关认可如何处理私人数据的更多信息，请单击[此处](#)。
4. 客户端应用程序将交易（包括带有私有数据哈希的投标响应）提交给排序服务。带有私有数据哈希的事务通常包含在块中。具有私有数据哈希的块将分配给所有对等方。这样，通道上的所有对等方都可以以一致的方式用私有数据的哈希值验证事务，而无需知道实际的私有数据。
5. 在块提交时，授权对等方使用收集策略来确定他们是否被授权有权访问私有数据。如果这样做，他们将首先检查其本地，以确定他们是否在链码认可时已收到私有数据。如果没有，他们将尝试从另一个授权对等方获取私有数据。然后，他们将根据公共块中的哈希来验证私有数据，并提交事务和该块。验证/提交后，私有数据将移至私有状态数据库和私有写入集存储的副本。然后从中删除私人数据。

`transient data store``transient data store`

5. 清除私人数据

对于非常敏感的数据，即使共享私有数据的各方也可能希望（或政府法规可能要求）定期“清除”其对等方的数据，在区块链上保留数据的哈希值以作为对私人数据。

在某些情况下，私有数据仅需要存在于对等方的私有数据库中，直到可以将其复制到对等方区块链外部的数据库中为止。数据也可能只需要存在于对等方上，直到完成链码业务流程（交易结算，合同履行等）。

为了支持这些用例，如果尚未针对可配置的块数进行修改，则可以清除私有数据。无法从链码中查询已清除的私有数据，并且其他请求对等方也无法使用。

6. 如何定义私人数据收集

有关集合定义的更多详细信息以及有关私有数据和集合的其他低级信息，请参考[私有数据参考主题](#)。

渠道能力

受众群体：频道管理员，节点管理员

注意：这是高级 Fabric 概念，新用户或应用程序开发人员无需了解。但是，随着渠道和网络的成熟，理解和管理功能变得至关重要。此外，重要的是要认识到，更新功能是升级节点的不同过程，尽管通常是相关的。我们将在本主题中对此进行详细描述。

因为 Fabric 是一个分布式系统，通常将涉及多个组织，所以可能（通常）不同版本的 Fabric 代码将存在于网络中的不同节点以及该网络的通道中。Fabric 允许这样做-不必每个对等方和排序节点都处于同一版本级别。实际上，支持不同的版本级别可以实现 Fabric 节点的滚动升级。什么是重要的是，网络和渠道以同样的方式处理事情，创造之类的东西通道配置更新和 chaincode 调用确定的结果。没有确定性的结果，一个通道上的一个对等方可能会使交易无效，而另一对等方可能对其进行验证。

为此，Fabric 定义了所谓的“功能”级别。这些功能在每个通道的配置中定义，通过定义行为产生一致结果的级别来确保确定性。您将看到，这些功能的版本与节点二进制版本密切相关。功能使运行在不同版本级别的节点能够以兼容且一致的方式运行，并且要在特定块高度使用给定的通道配置。您还将看到功能在配置树的许多部分中存在，这些功能是根据特定任务的管理定义的。

如您所见，有时需要将您的频道更新到新的功能级别以启用新功能。

1. 节点版本和功能版本

如果您熟悉 Hyperledger Fabric，则会知道它遵循典型的语义版本控制模式：v1.1，v1.2.1 等。这些版本指的是发行版及其相关的二进制版本。

功能遵循相同的语义版本约定。有 v1.1 功能和 v1.2 功能等等。但重要的是要注意一些区别。

- 每个发行版不一定都具有新的功能级别。**建立新功能的需要视具体情况而定，并且主要取决于新功能和较旧的二进制版本的向后兼容性。例如，在 v1.4.1 中添加 Raft 排序服务不会改变处理交易或排序服务功能的方式，因此不需要建立任何新功能。[私人数据](#)另一方面，v1.2 之前的对等节点无法处理，因此需要建立 v1.2 能力级别。由于并非每个版本都包含改变交易处理方式的新功能（或错误修复），因此某些版本将不需要任何新功能（例如 v1.4），而其他版本仅具有特定级别的新功能（例如作为 v1.2 和 v1.3）。稍后我们将讨论功能的“级别”以及它们在配置树中的位置。
- 节点必须至少处于通道中某些功能的级别。**当对等方加入通道时，它将顺序读取账本中的所有模块，从通道的创始模块开始，一直到交易模块和任何后续配置模块。如果某个节点（例如对等节点）尝试读取包含对其不了解的功能的更新的块（例如，v1.2 对等节点尝试读取具有 v1.4.2 应用程序功能的块），则该对等节点会崩溃。这种崩溃行为是有意为之的，因为 v1.2 对等端无法验证或提交任何超过此时间点的服务。加入频道之前，**确保该节点处于与该节点相关的通道配置中指定的功能的 Fabric 版本（二进制）级别或更高级别。**稍后我们将讨论哪些功能与哪些节点相关。但是，由于没有用户希望其节点崩溃，因此强烈建议在尝试更新功能之前将所有节点更新到所需级别（最好更新到最新版本）。这与默认的 Fabric 建议一致，即 **始终**处于最新的二进制和功能级别。

如果用户无法升级其二进制文件，则必须将功能保留在较低级别。较低级别的二进制文件和功能仍将按预

期工作。但是，请记住，即使用户选择不更新其功能，也总是始终更新到新的二进制文件是最佳实践。由于功能本身还包含错误修复，因此始终建议在网络二进制文件支持功能后对其进行更新。

2. 功能配置分组

正如我们前面所讨论的，没有一个能力级别涵盖整个通道。而是具有三种功能，每种功能代表一个管理领域。

- **排序节点**: 这些功能管理排序服务专有的任务和处理。由于这些功能不涉及影响交易或对等方的流程，因此更新它们完全由排序服务管理员负责（对等方无需了解订购方功能，因此无论将其更新为何种订购方功能，都不会崩溃）。请注意，这些功能在 v1.1 和 v1.4.2 之间没有变化。但是，正如我们将在**通道**部分中看到的那样，这并不意味着 v1.1 排序节点将在功能级别低于 v1.4.2 的所有通道上工作。
- **应用程序**: 这些功能管理对等方专有的任务和处理。由于排序服务管理员在确定对等组织之间的交易性质方面没有任何作用，因此更改此功能级别完全取决于对等组织。例如，只能在启用了 v1.2 应用程序组功能（或更高版本）的通道上启用私有数据。对于私有数据，这是唯一必须启用的功能，因为私有数据的工作方式不需要更改渠道管理或排序服务处理交易的方式。
- **渠道**: 此分组包含由对等组织和排序服务**共同管理**的任务。例如，此功能定义了处理通道配置更新的级别，该级别由对等组织发起并由排序服务编排。在实际级别上，**此分组定义了通道中所有二进制文件的最低级别，因为排序节点和对等节点都必须至少处于与该功能相对应的二进制级别，才能处理该功能。**

在**排序**和**通道**通道的功能是由从订货系统的信道，其中修改它们是排序服务管理员所独有的领域默认继承。因此，对等组织应在将其对等组织加入该渠道之前检查该渠道的起源。尽管通道功能由排序节点在排序节点系统通道中进行管理（就像联盟成员身份一样），但通常并期望订购管理员将与联盟管理员进行协调，以确保仅在联盟存在时才升级通道能力准备好了。

由于排序系统通道未定义**应用程序**功能，因此在创建通道的创始模块时必须在通道配置文件中指定此功能。有关创建通道的[创世纪块](#)的更多信息，请查看 [configtx](#)。

在指定或修改应用程序功能时**请小心**。由于排序服务无法验证功能级别是否有效，因此即使没有这样的功能，它也将允许创建（或修改）通道以包含例如 v1.8 应用程序功能。如我们所显示的，任何尝试读取具有此功能的配置块的对等方都会崩溃，即使有可能再次将通道修改为有效功能也没有关系，因为任何对等方都无法获得越过具有无效 v1.8 功能的块。

要全面了解当前有效的排序节点、应用程序和渠道功能，请查看示例 [configtx.yaml](#) 文件，该文件在“功能”部分列出了这些[文件](#)。

有关功能及其在通道配置中的位置的更多特定信息，请查看[定义功能需求](#)。