

## 第四章 Fabric简介

- 超级账本（Hyperledger）项目是一个旨在推动区块链跨行业应用的开源项目，于2015由Linux基金会主导发起，成员包括金融，银行，物联网，供应链，制造和科技行业的领头羊。

# Fabric简介



- **Hyperledger Fabric**是一个开源的、企业级的并且基于许可模式的分布式账本平台
- **Fabric**专门为企业级应用环境而设计，实现了一个高度模块化和可配置的体系结构
- 允许不同行业基于不同的行业用例对**Fabric**进行扩展、创新、优化，建立不同的行业应用模块和平台

# Fabric简介 — 框架图



# Fabric简介 — 特性



- 模块化框架
  - 核心模块具有高度模块化，都可通过更改配置文件加载不同的模块实现，不同的行业可以开发或扩展基于本行业用例的模块实现
- 基于许可的区块链
- 新的交易模型
- 企业级智能合约
- 隐私和保密

# Fabric简介 — 模块化特性



- 可插拔的共识模块
  - 目前Fabric 1.3版本已经实现solo模式、Kafka模式
  - 开发基于拜占庭容错（BFT）的共识协议
- 可随意安装部署的智能合约
  - Fabric中的智能合约称之为链码（chaincode）
  - 链码可以动态安装与升级，从而扩展平台的业务逻辑
- 可插拔的背书和验证系统链码
  - 可以给每个通道的每个链码指定不同的背书和验证系统链码，从而完成个性化的验证要求

# Fabric简介 — 模块化特性



- 可选的对等网络gossip服务
  - 传播来自共识服务所形成的区块
- 不同形式的账本结构
  - Fabric的账本状态数据可以使用不同的数据库管理系统
  - 默认数据库是LevelDB，也可以配置成第三方数据库，如CouchDB。
- 可插拔的成员服务提供者
  - 主要负责将网络中的某个实体（节点、组织、成员）和一个加密身份相关联，对实体身份进行认证，是权限管理的基础

# Fabric简介 — 基于许可的区块链



- 经过认证、审查才能参与到区块链的交易中
- 参与者在一种具有一定程度信任的治理模式下相互协作
- 可以使用更传统的崩溃容错（CFT）的共识协议，从而避免昂贵的POW等共识协议
- 许可方法
  - 颁发交易参与方、节点等实体的身份证书和用户的身份证书；
  - 交易经过签名、背书，证明了其合法性。

# Fabric简介 — 新的交易模型



- 传统区块链运行于“共识-运行”框架下
  - 所有节点顺序执行交易，导致性能和规模受限
  - 智能合约要在系统中每个节点执行，要求系统采取复杂措施来保护整个系统免受恶意合同的影响，从而限制了系统弹性
  - 如智能合约存储于账本中，限制了智能合约的规模。
- 交易运行步骤
  - 通过一个共识协议对一组交易形成共识区块；
  - 每个节点顺序运行共识区块中的交易



# Fabric简介— 新的交易模型



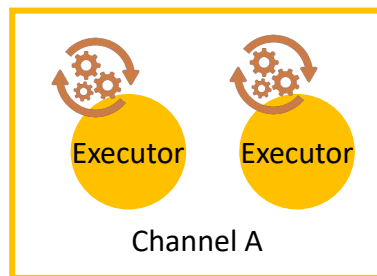
- “运行-共识-验证”（**execute-order-validate**）交易执行框架
- 在Fabric 1.0中实现，有**客户端**、**背书节点**和**排序节点**三种角色；
  - **客户端**发送交易；**背书节点**执行和验证交易；**排序节点**通过共识确定一批交易的全局顺序。



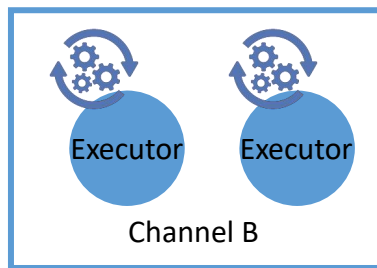
客户端



客户端

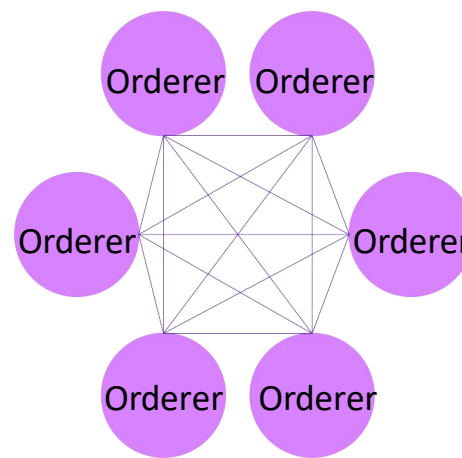


Channel A



Channel B

执行节点

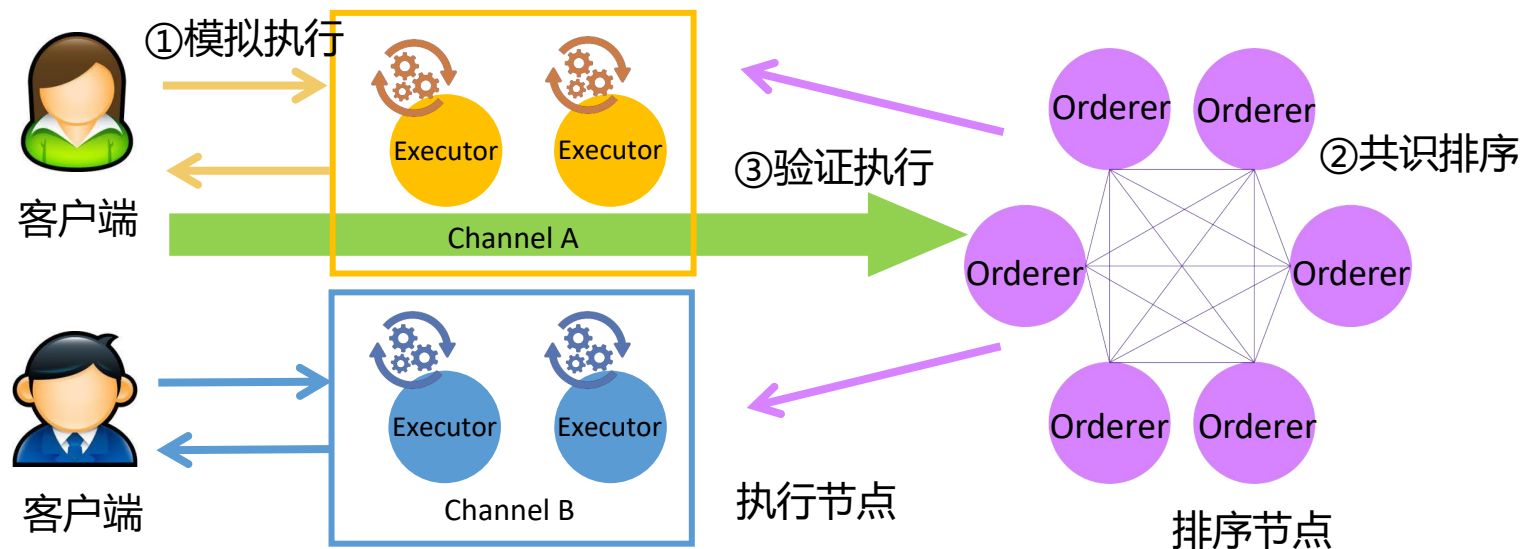


排序节点

# Fabric 简介—新的交易模型



- 交易运行过程分为三个步骤：
  - 交易在各channel之间并行模拟执行，但执行结果不写入；
  - 共识节点对多channel提交的交易进行共识排序；
  - 各节点验证该共识顺序是否破坏模拟执行的可串行性。



# Fabric简介—新的交易模型



- 运行交易的节点是智能合约要求的有限的几个节点
- 在取得共识前运行智能合约，使得账本仅需保存智能合约的运行结果而不是智能合约本身
- 智能合约可以以文件的形式部署在需要的节点中

# Fabric简介——企业级智能合约



- 智能合约在Fabric中称为链码（**chaincode**），是区块链应用中的业务逻辑。
- 基于链码的背书策略指定哪些节点以及多少节点需要保证智能合约的正确运行
  - 每个交易仅需背书策略所必需的节点的子集来认可交易执行结果
  - 这使得Fabric在运行阶段能够过滤掉不一致的结果，从而消除不确定性。
- 运行不确定性的消除使得Fabric成为第一个支持使用通用语言（如Java、Go和Node.js）开发智能合约的区块链平台

# Fabric简介——隐私和保密



- Fabric在数据加密、数字签名、安全网络传输的基础上，针对企业应用进行了如下设计：
  - 智能合约的保密要求
    - 实例：合同的签署各方能够看到优惠的费率及交易的金额
    - 通过将智能合约部署在特定的背书节点，从而保证智能合约仅对特定的交易方可见，从而确保了智能合约的私密性。[物理隔离]
  - 账本的保密要求
    - 传统的区块链平台中，账本数据经过加密处理，但账本数据部署在所有的节点，经过足够的时间或者资源，这些数据还是能被破解，导致企业敏感数据的泄密
    - Fabric中引入通道（channel）的概念，有关交易参与方建立一个通道，账本数据仅仅保存在参与通道的相关节点中[物理隔离]

# 核心概念

- 主要介绍Fabric中的几个重要术语。

# 核心概念



- 交易
- 背书
- 链码 (chaincode)
- 通道 (channel)
- 排序服务

# 核心概念 — 交易



- Fabric中交易是链码的一次调用，根据链码完成的业务逻辑完成不同的功能，可以实现对账本状态的改变，也可以是一次对账本数据的查询。
- 部署交易（**Deploy transactions**）
  - 主要完成新建链码的安装和初始化，部署交易执行成功也就意味着一个新的链码已经准备好执行对链码调用的交易；
- 调用交易（**Invoke transactions**）
  - 调用已部署的链码的指定的方法，返回方法的执行结果。
  - 部署交易是调用交易的特例，其参数是打包的链码。
  - 一个完整的交易执行过程需要经过提案、背书、共识（排序）最终提交到账本中。



# 核心概念 — 背书



- 背书在商业上是指对某种交易行为进行担保、保证
- Fabric 借签这一流程，在交易进入排序和计入账本之前需要交易的相关各方进行背书
  - Fabric 中背书是指背书节点对收到的客户端的交易提案按照自身逻辑进行检查，并调用链码执行交易，背书节点对请求的提案和链码执行结果（状态变更有关的读写集合）进行数字签名的过程。
  - 对于调用链码的应用来说，需要根据链码的要求进行背书才会认为合法，才能提交给排序节点。

# 核心概念 — 背书



- 在Fabric中链码的背书要求称之为背书策略
  - 背书策略可以要求指定成员集合中成员的一致同意或者一部分成员的同意甚至某个成员的支持，可以使用多种规则组合。

# 核心概念 — 链码 (chaincode)



- Fabric中智能合约称为链码 (Chaincode)，链码是使用编程语言实现预定义接口 (Chaincode接口) 的一段应用程序，支持Go、Node.js、Java等高级编程语言
- 链码分为用户链码和系统链码
  - 用户链码运行于独立的容器中，通常所说的链码是用户链码即开发人员基于企业用例所开发的链码；
  - 系统链码则完成链码生命周期管理、分布式账本查询、背书签名、提交验证等系统功能。

# 核心概念 — 通道 (channel)



- 通道从狭义讲是Fabric网络成员中一部分节点构成的一个专用通信通道，从而限制网络中其他成员的访问。
- Fabric中通道还包括绑定在通道上的交易、链码、共享账本、成员节点、排序节点等配置和数据。
- 加入通道的节点要有MSP赋予的唯一身份标识，并通过MSP认证通道的相关节点及服务。

# 核心概念 — 通道 (channel)



- Fabric把通道分为两类系统通道 (System Channel) 和应用通道 (Application Channel)
  - 系统通道则主要负责对应用通道进行管理
  - 应用通道主要运行用户链码交易
- 创始通道 (Genesis Channel)
  - 初始Fabric网络启动时对于每个排序服务来说, 启动的一个特殊的排序系统通道
  - 该通道绑定排序节点并负责应用通道的创建, 作为系统创建的首个通道

# 核心概念 — 应用通道创建



- 向系统通道发送配置交易（**Configuration Transaction**）
- 排序服务会为该应用通道创建一个创始区块（**Genesis Block**），创始区块中包含该配置交易及初始配置的相关信息，该配置信息包括通道基本信息、通道访问策略、初始包含的成员信息、锚节点、排序服务地址等。
- 当某个节点申请加入某条通道时，获取指定通道的配置区块，并调用系统链码的**JoinChain**方法获取创世区块，并完成账本、通道相关数据结构的初始化工作。

# 核心概念 — 排序服务



- Fabric的排序服务接收来自客户端的背书后的交易并按通道进行全局一致的排序，将一段时间内的交易形成区块并广播给相应的通道。
- 排序服务就是Fabric提供共识服务的模块，以可插拔的形式实现。
  - 目前Fabric除了测试的solo模式，还包括已商业应用的Kafka模式
  - 第三方的CFT或者BFT共识算法也可以插件的形式包含在Fabric中。
- Fabric排序服务除对交易进行排序形成共识区块或者创建用户通道外并不执行其他操作，也可以通过多个排序节点扩展性能。

# 区块链数据结构

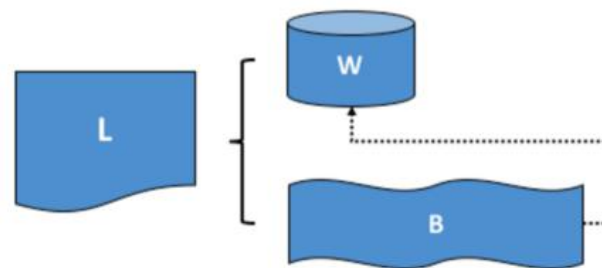
- Fabric中分布式账本的数据结构简介








# 区块链数据结构 — 账本 (Ledger)



- 区块链 ( blockchain ) 结构
  - 记录发生在网络中的交易信息及交易所引起的状态变更。
- 状态数据库 (State Database)
  - 由区块链结构中交易执行的结果生成，记录最新的世界状态；
  - (K, V, S): 具有版本号的键/值序列对
- 历史数据库 (History Database)
  - 存放各个状态的历史变化记录；



	Ledger
	World State
	Blockchain
	L comprises B and W
	B determines W

# 区块链数据结构 — 账本 (Ledger)

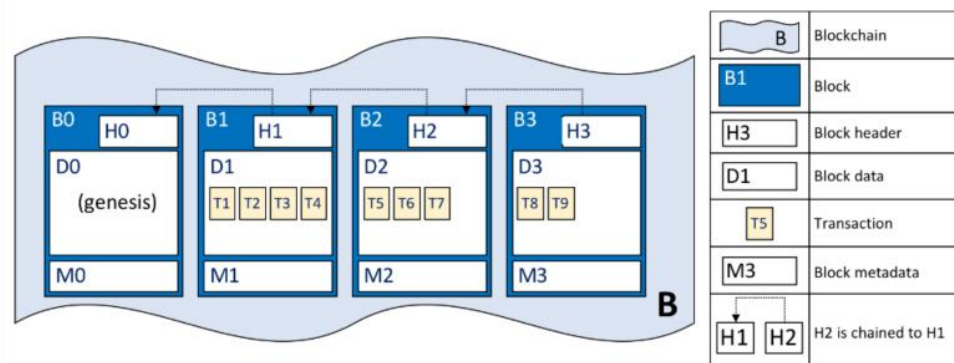


- 一般区块链结构通过文件系统进行存储
  - 从数据库角度讲，区块链结构保存了状态变更的记录
  - 变更记录中保存了本次交易涉及的变更前状态数据（读集合）和变更后状态数据（写集合）
- 状态数据库则支持可插拔模块，可支持LevelDB、CouchDB，目前Fabric默认支持LevelDB。
  - 从数据库角度讲，状态数据库则保存了最新的状态信息

# 区块链数据结构 — 区块链结构



- 区块链的第一个区块是创世区块
- 在Fabric中并不保存用户交易，主要保存通道创建时的通道配置交易，以后通道配置信息的更改会有新的配置区块来代替。
- 链式的结构连接所有区块，经过排序服务达成共识的最新区块，会根据区块编号追加到区块链的末尾，每个区块中包含了顺序执行的交易，这些交易可能成功也可能失败。



# 区块链数据结构 — 区块



- 区块头部

- 区块编号，包含一个数字区块编号，从0开始递增，最新的区块追加到链的尾部；
- 当前区块Hash值，即，当前区块包含所有交易的散列；
- 前一个区块Hash值，通过该值所有区块形成链状结构。

- 区块数据

- 按照排序服务形成顺序排列的交易列表。

- 区块元数据

- 包括块的写入时间及写入者的签名、公钥和证书等信息。
- 提交节点提交区块时，还为每个交易添加是否有效的指示标志
- 该指示标志不包含在区块Hash值（区块创建时生成）中

# Fabric框架

- Fabric是一个有许多节点构成的分布式系统，节点运行链码（智能合约）、保存状态和账本数据、执行交易等功能。

# Fabric框架 — 整体架构



# Fabric框架 — 整体架构



- Fabric对外提供了基于gRPC的API以及封装API的相应语言的SDK，操作员也可以通过命令行（CLI）直接执行Fabric支持的命令。企业应用通过SDK或API接口访问Fabric区块链网络的多种资源，包括账本、交易、链码并可进行权限管理。
- 应用通过交易向账本记录数据，改变状态，交易的执行逻辑则由链码来完成。企业应用依据Fabric区块链中的事件触发企业应用中的后续处理流程，从而完成基于事件驱动的异步模型。
- 交易、共识服务、分布式账本、状态数据库形成Fabric的区块链服务体系，交易、链码和容器形成Fabric的智能合约服务体系。

# Fabric框架 — 整体架构



- 成员服务提供者则是Fabric的安全认证体系，包括用户管理、权限管理、证书管理等；
- 区块链加密服务提供者是Fabric以可插拔的方式提供的加解密算法等；
- 底层则是由多个节点构成的P2P网络，通过基于gRPC的通道进行交互，利用Gossip协议进行区块链同步。

*通过层次化、模块化的框架结构以及可插拔的模块支持，允许企业开发行业区块链平台。*



# Fabric框架 — 节点网络



- 节点的概念来自于P2P分布式网络，是在网络中担任一定职能的服务或软件，节点功能可能是对等一致的，也可能是分工合作的。
- Fabric中的节点是区块链网络中的通信主体，一个节点也是一组逻辑功能的逻辑表示，因此一个物理服务器中可以运行不同类型的节点。
- 在部署时需要考虑如何对节点进行分组并对节点进行访问控制，从而形成一个高效、安全的节点网络。

# Fabric框架 — 节点网络



- 客户端

- 客户端代表最终的用户实体，可以是命令行客户端，也可以是一个应用程序，当作为应用程序时，通过Fabric提供的SDK和区块链网络进行交互。
- 客户端通常会同时连接到一个节点和一个排序节点，客户端创建并发送交易给相关背书节点，符合背书要求后，发送交易给排序节点。

- 节点 (Peer)

- 维护节点网络，保持与节点中其他网络的连接；
- 同步账本数据；
- 节点接收排序节点的新的共识块，更新账本和状态数据库。
- 节点分为：提交节点 (Committer) 和背书节点 (Endorser)

# Fabric框架 — 节点网络



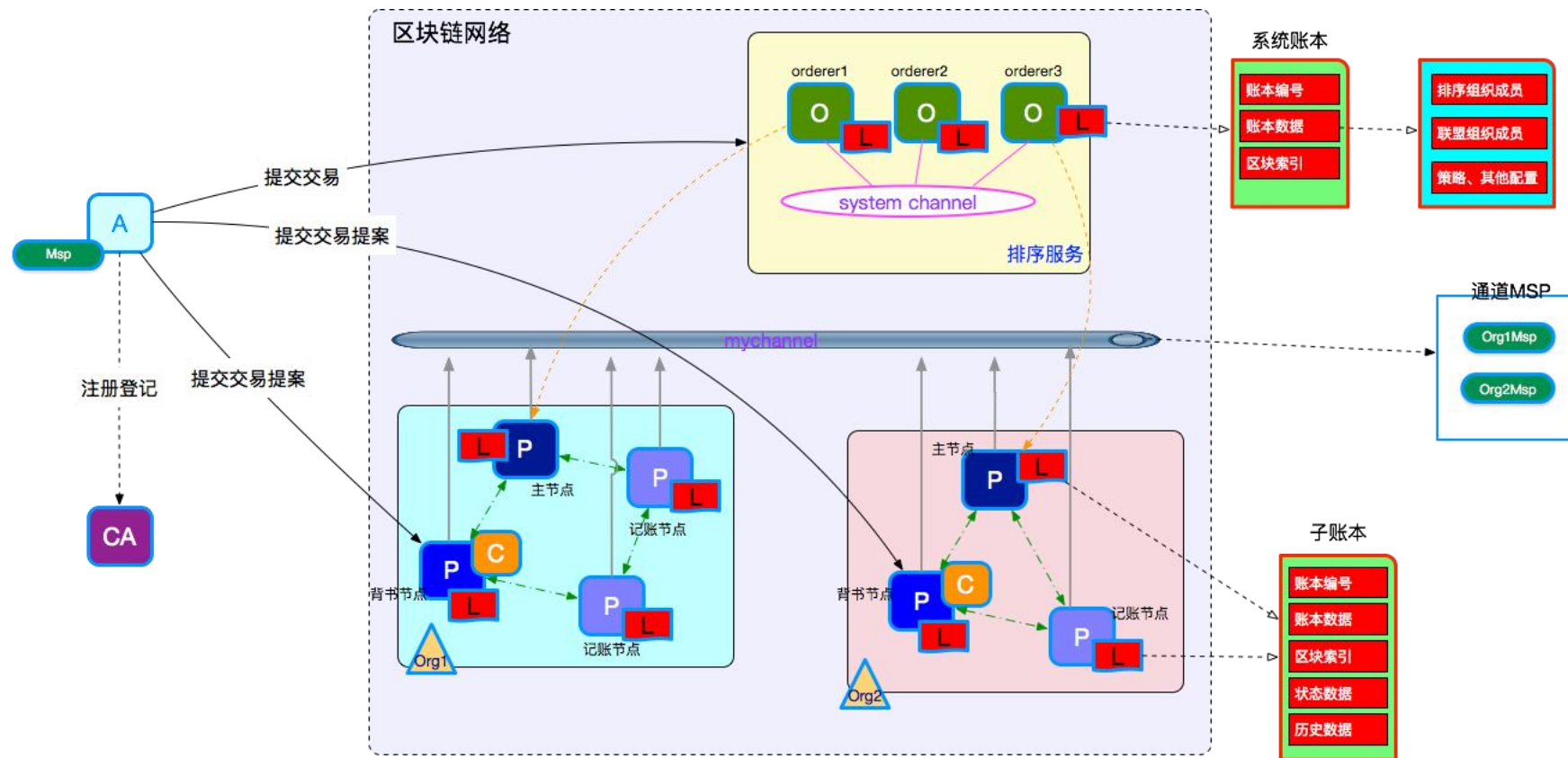
- 提交节点 (Committer)
  - 对来自排序节点的新共识块进行验证
  - 记入账本和状态数据库
  - Fabric中的所有节点都包含该逻辑功能，也可以说所有节点都是提交节点
- 背书节点 (Endorser)
  - 执行链码并对执行的结果进行背书
  - 每个链码的背书节点由链码背书策略决定，不同链码对应的背书节点也可能是不同的。

# Fabric框架 — 节点网络



- 排序服务节点或称为排序节点
  - 排序服务节点接收包含背书签名的交易
  - 对未打包的交易进行排序生成区块
  - 广播给Peer节点
- CA节点
  - CA节点是可选的，它主要作为证书颁发机构，也可以用其他成熟的第三方CA颁发证书。

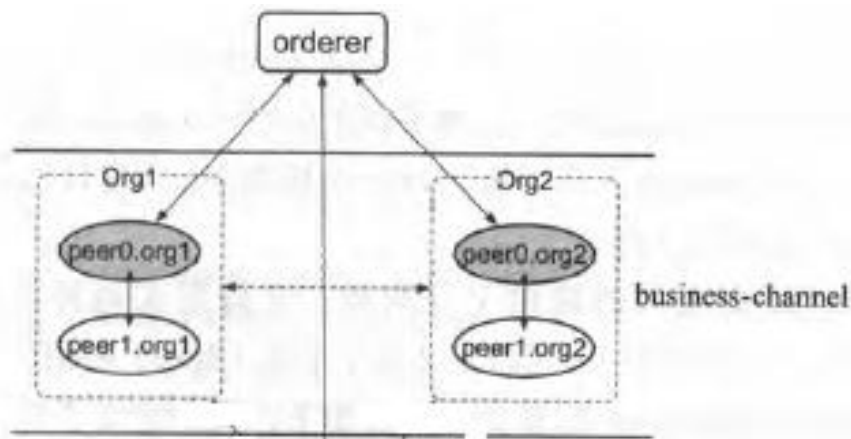
# Fabric框架 — 节点网络



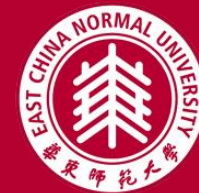
# Fabric框架 — 节点网络



- 一个Fabric网络中可以有多多个组织（成员），一个组织在一个通道上可以有多个节点
- 主节点（**Leader Peer**）作为代表和排序服务节点通信，负责从排序服务节点处获取最新的区块并在组织内部同步
- 锚节点（**Anchor Peer**）代表组织与其他组织成员进行信息交换。



# Fabric框架 — 节点网络



- 为保证私密性Fabric区块链网络引入通道（channel）的概念，一个通道可以认为是区块链的一个虚拟专用子网（类似于VPN）。
- 区块链网络初始建立时系统默认启动一个包含所有排序节点的系统通道，组织成员根据需要需要通过系统通道创建专有通道，一个节点可以加入一个或多个通道，一个通道也可以包含多个节点。
- Fabric基于通道提交交易、保存账本、执行链码、保存状态，不同通道的账本、状态和智能合约、策略都是隔离的。

# 交易流程

- 一个转账交易经过一些列步骤就不可更改的写入了区块链数据库，参与各方可以查询相关交易数据。

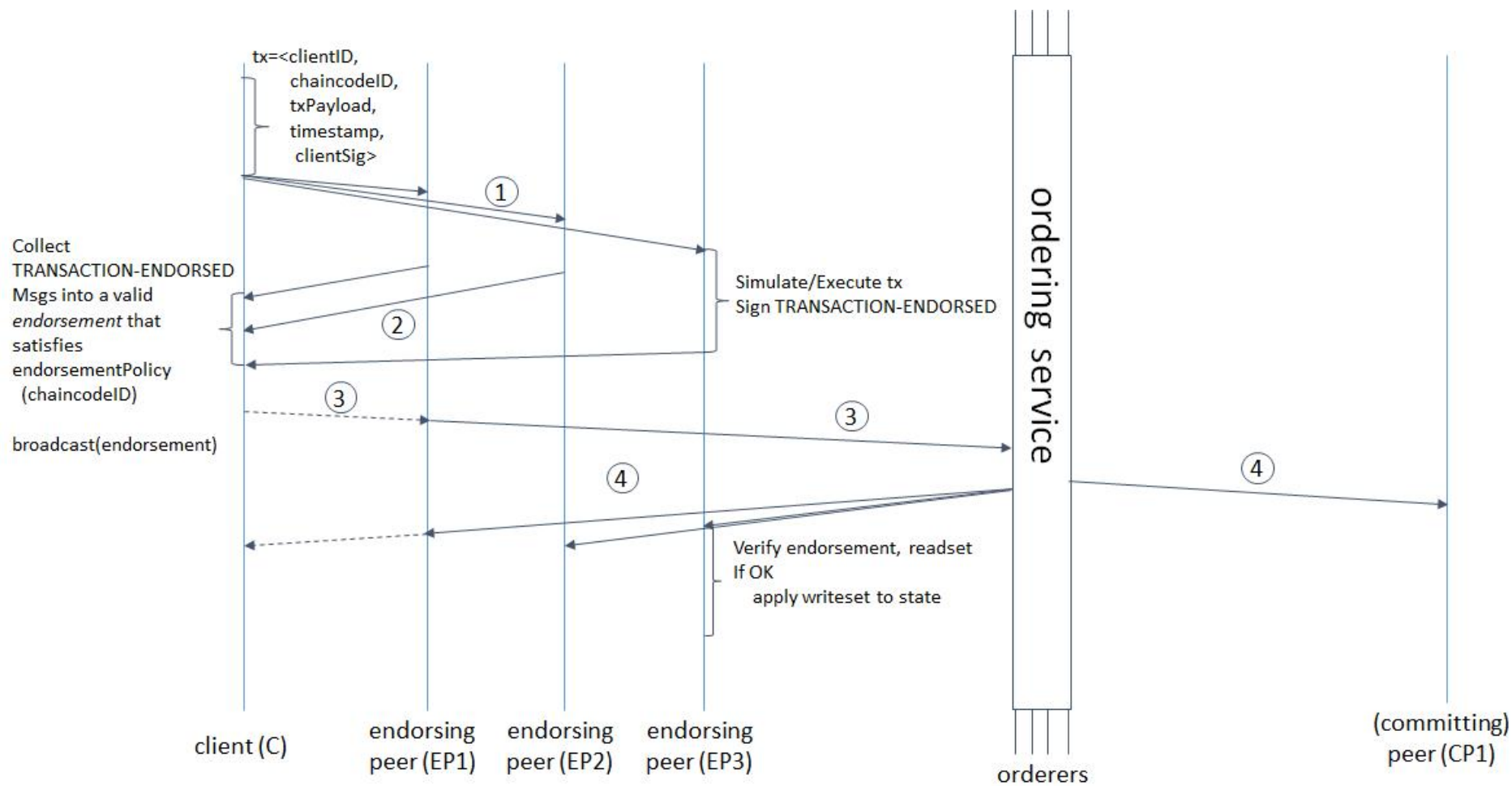


# 交易流程 — 应用场景



- 应用场景：企业A向B支付货款
- 节点网络：
  - 假设企业A和B都有一个节点（节点A和节点B）连接到网络
  - 组织A和B的节点都连接到一个存在的通道
- 安全体系
  - 应用程序的用户具有CA签发的数字证书并有相应的权限
- 链码
  - 链码已经安装到Fabric系统并绑定到通道
  - 链码的背书策略要求转账交易必须有节点A和节点B的背书

# 交易流程 — 时序图



# 交易流程 — Step 1



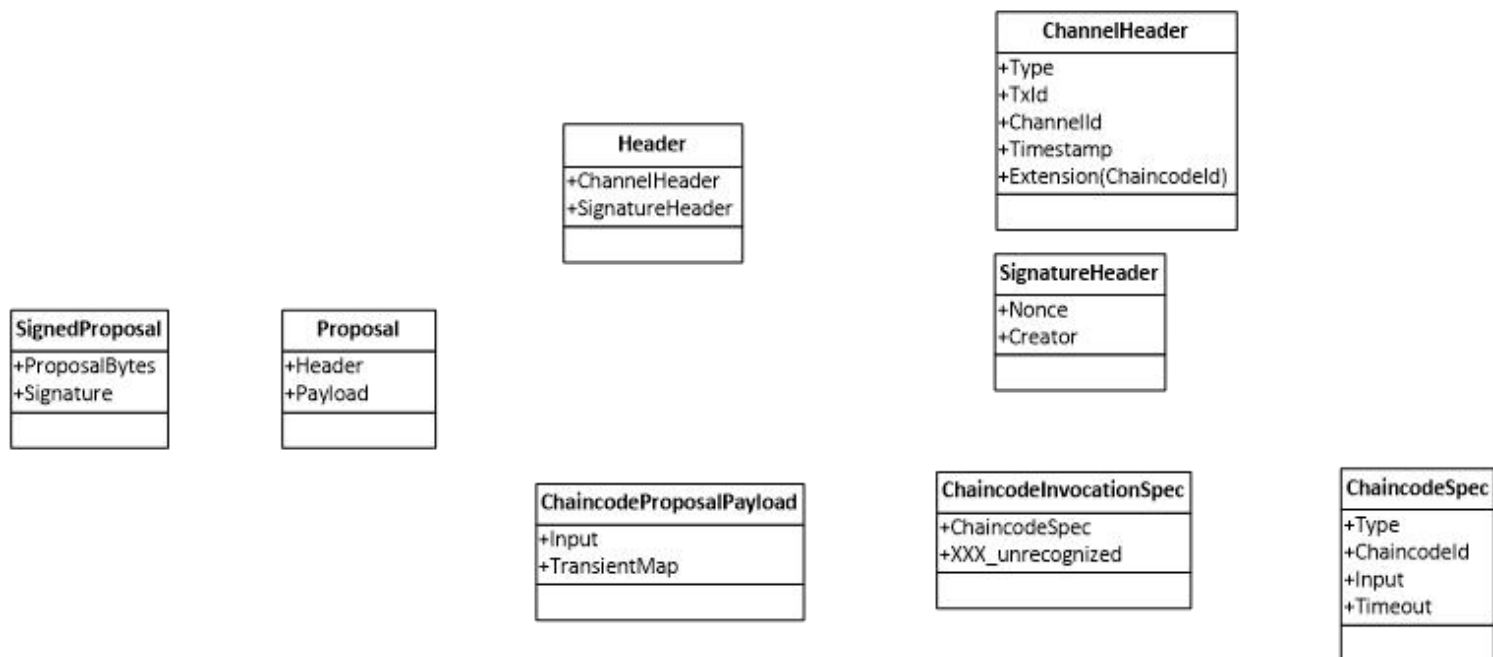
*客户端创建一个交易提案并根据背书策略发送给相应的背书节点*

- 创建ChaincodeInvocationSpec结构，主要包括ChaincodeId和链码方法名及相关参数；
- 计算交易Id（Transaction Id）；
- 创建Proposal结构，通道头部信息中指定交易类型HeaderType\_ENDORSER\_TRANSACTION，设置通道Id、交易Id，设置签名头部的随机数和交易创建者；
- 使用创建者的数字证书对Proposal结构进行签名，形成SignedProposal结构。

# 交易流程 — Step 1



- 根据链码的背书策略获取需要提交的背书节点；
- 把签名后的交易提案（**SignedProposal**）提交给相应的背书节点运行。调用**EndorserClient**接口的**ProcessProposal**方法。



# 交易流程 — Step 1



- **ChaincodeProposalPayload**中的**TransientMap**保存针对应用层的私密数据[Private Data]
- 该部分数据经过特殊加密或者Hash，仅仅供授权的组织查看
- 这部分数据在节点中单独保存且仅保存在授权的节点中
- 排序服务不会看到也不会对这部分数据进行处理

*该部分的代码实现可以参考[fabric/peer/chaincode/invoke.go](https://github.com/hyperledger/fabric/blob/master/peer/chaincode/invoke.go)*

# 交易流程 — Step 2



背书节点收到*SignedProposal*消息后调用*EndorserServer*接口的*ProcessProposal*方法处理签名的交易提案。

- 对签名提案进行检查和校验，具体内容包括：
  - 消息结构是否完整；
  - 消息是否已经被提交过，从而避免重发攻击；
  - 客户端签名是否有效，主要是调用MSP的相关服务来完成；
  - 提交交易的客户端是否具有相应通道的权限，如读写权限。

# 交易流程 — Step 2



- 启动仿真模式，构造链码**Invoke**参数，并调用链码相关方法。
  - 链码基于当前状态数据库执行链码并返回响应消息，包括响应值、读集合、写集合，在这个步骤不涉及任何对状态数据库和账本的更改；
- 调用相应的**ESCC**链码，对交易结果进行背书，最终形成**ProposalResponse**结构
  - 包含Payload（Response的读写数据集）、Endorsement（背书信息）。
- 背书响应及相关的事件返回客户端

# 交易流程 — Step 3



*客户端收集足够的提案背书响应，并向排序服务广播具有足够背书的交易*

- 若该交易仅仅查询账本、交易、状态等信息，客户端仅需解析响应消息不需要把交易提交给排序服务。
- 若该交易涉及账本数据变更，则需向排序服务提交交易，这时客户端需确保提案响应的Payload属性是一致的即交易执行的结果是一致的。
- 客户端把提案、响应及背书信息打包形成一个完整的交易(Transaction)，添加头部及客户端签名后形成Envelope，最后通过BroadcastClient向排序服务广播该Envelope。
- 在该阶段，客户端会过滤掉TransientMap等受限访问的数据，这些数据也不会写入区块链数据库。



# 交易流程 — Step 4



## 排序服务形成最新的共识块

- 排序服务接收到来自客户端或代理节点广播的交易消息，这些消息可能来自不同的通道
- 排序服务按照通道根据可插拔的共识算法，创建每个通道的最新共识块
- 排序服务形成共识块后把共识块交付给各个组织的主节点
- 主节点收到共识块后通过Gossip协议向本组织的其他节点传播，最终一个通道中所有节点都能收到最新的共识块。

# 交易流程 — Step 5



## 验证和提交交易到账本数据库

- 节点接收到共识块后，对块中的所有交易进行验证
  - 检查交易的数字签名
  - 交易符合背书策略
  - 交易仿真运行时的读取集合版本自运行以来没有变化
- 在提交阶段检查，确保客户端即使没有对提案响应进行分析并检查背书策略，也能在提交阶段过滤掉不符合要求的交易。
- 在验证阶段，如果一笔交易中每个读集中的键与世界状态中对应键的版本号一致，那么该笔交易被认为是有效的。

# 交易流程 — Step 6



## 更新账本数据库

- 节点把验证过的区块附加到相应通道的区块链中
- 对于区块中验证通过的交易，把写集合提交到状态数据库
- 向客户端发出一个事件用于通知交易已经写入账本或者交易验证无效，客户端收到事件后完成后续的业务处理。

# 权限管理与策略

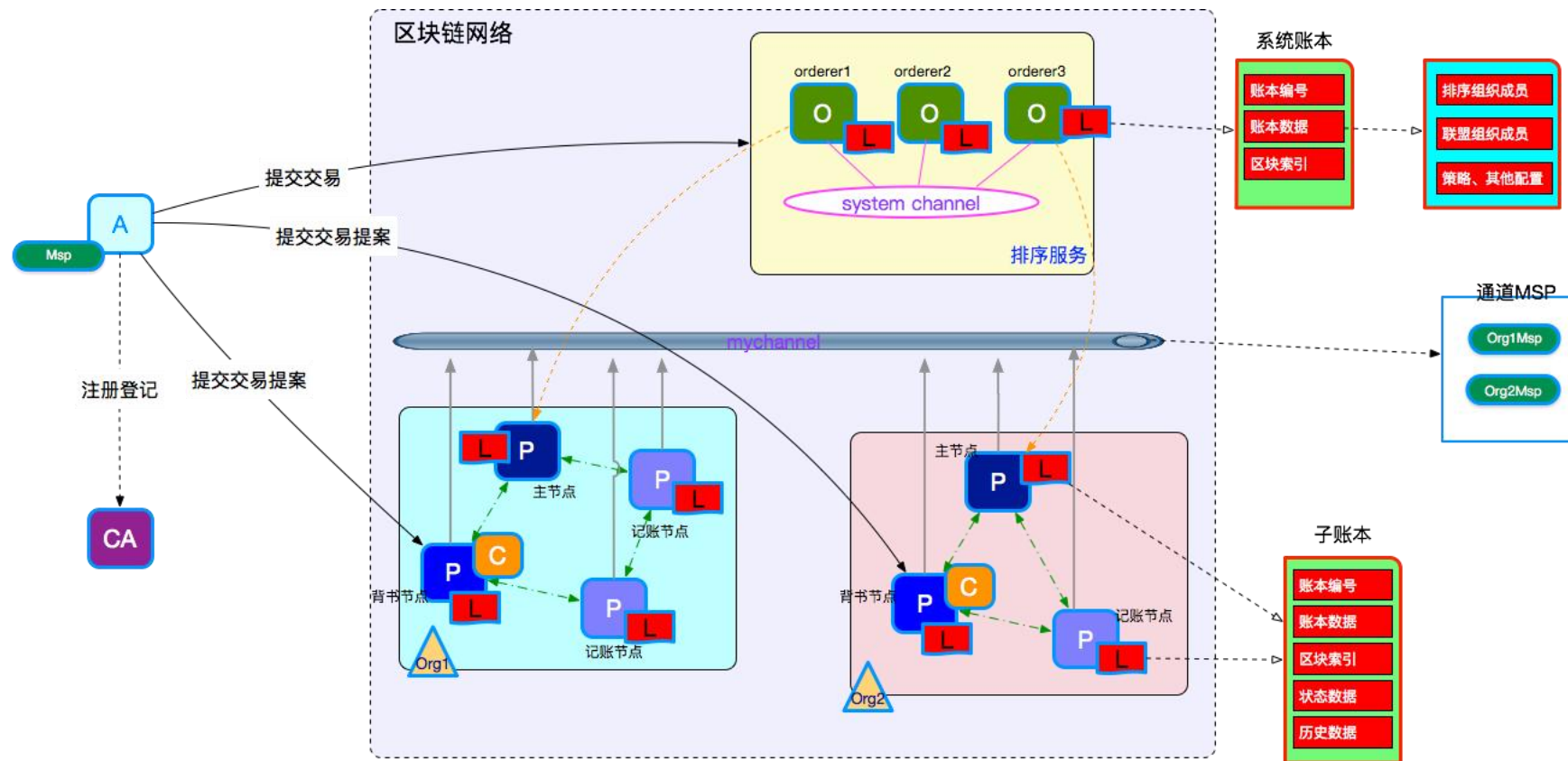
- 作为受限访问的区块链系统，通过相关的权限管理与策略确保系统的安全性。

# 权限管理与策略

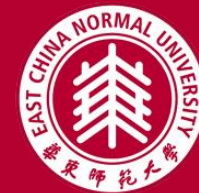


- 组织体系
- 权限体系
- 证书体系
- 成员服务提供者 – MSP

# Fabric框架 — 节点网络



# 权限管理与策略 — 组织体系

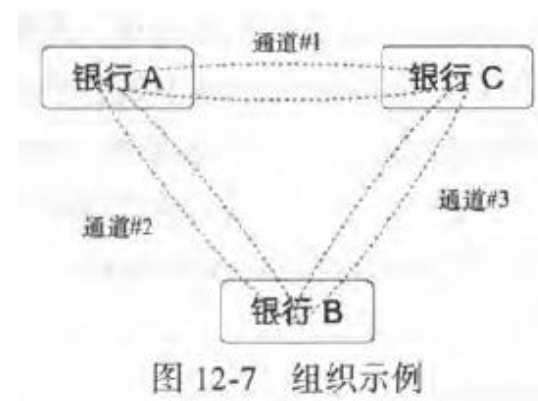


- 联盟（Consortium）

- 若干个组织构成的集合，联盟中的组织成员使用同一个排序服务；
- 联盟中的每个组织成员都有一个成员服务提供者（MSP）分配的ID信息。

- 组织（Organization）

- 代表一组拥有共同信任的根证书的成员（节点）
- 根证书是CA认证中心给自己颁发的证书，是信任链的起始点。
- 组织一般包括ID、名称、MSP信息、管理策略、认证密码库类型、锚节点位置等信息。



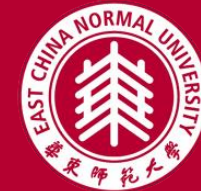
# 权限管理与策略 — 权限体系



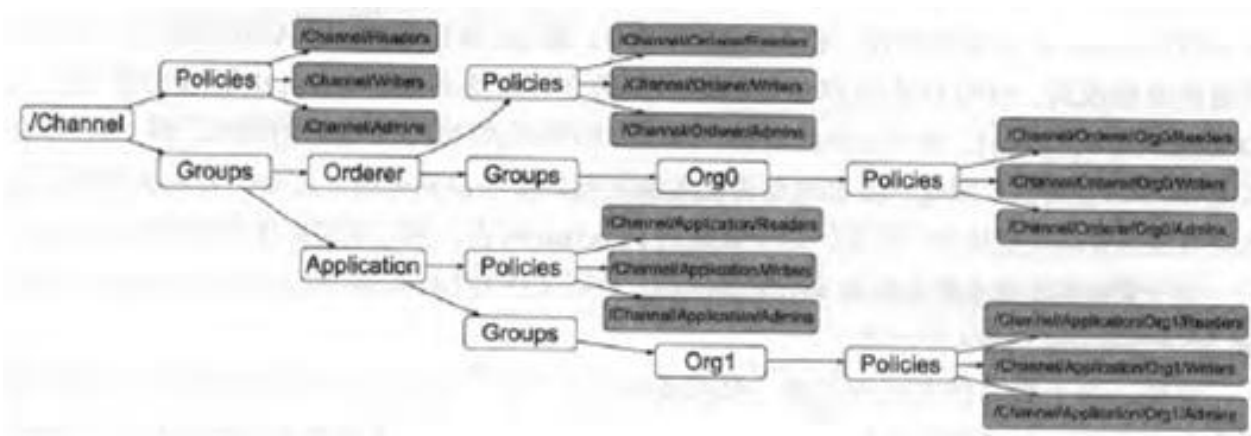
- 权限：管理员权限(admin)、读权限(read)和写权限(write)
  - 读权限（Readers）的操作限制包括获取通道的交易、区块等数据；
  - 写权限（ Writers）包括向通道发起交易等；
  - 管理员权限（Admins）包括加入通道、修改通道的配置信息等。
- 角色：管理员角色和普通成员角色
  - 管理员角色具有管理员权限
  - 普通成员角色则具有读或写权限



# 权限管理与策略 — 通道策略



- 在Fabric中**通道配置文件**中的**通道策略**负责对通道内各种操作权限进行管理。
  - 通道策略中会对管理员、读、写等权限定义具体允许的操作以及授权的用户列表。
  - 通道策略(Channel Policy)是层级化结构，最上层为/Channel，各级元素，下一级则挂接组织或组织单元



# 权限管理与策略 — 证书体系（组织）



以每个组织为单位，每个OU (*Organizational Unit*, 组织单位) 会生成单独的根证书。

- 组织的证书信息包括：

- CA：存放组织的根证书和对应的私钥文件，默认采用EC算法，证书为自签名。组织内的实体将基于该根证书作为证书根。
- MSP：存放代表该组织的身份信息，用于节点和客户端的MSP服务。
  - admincerts：组织管理员的身份验证证书，被根证书签名。
  - cacerts：组织的根证书，同ca目录下文件。
  - tlscacerts：用于TLS (Transport Layer Security, 传输层安全协议) 的CA证书，自签名。

# 权限管理与策略 — 证书体系（节点）



- Fabric中每个节点（背书节点、提交节点、排序节点）、客户端都会在相应目录中建立一套目录，用于存放相关的证书文件。该目录结构包括MSP 证书和TLS 证书两类。
  - MSP
    - admincerts：组织管理员的身份验证证书。Peer 将基于这些证书来认证交易签署者是否为管理员身份。
    - cacerts：存放组织的根证书。
    - keystore：本节点的身份私钥，用来签名。
    - signcerts：验证本节点签名的证书，被组织根证书签名。
    - tlscacerts：TLS 连接用的身份证书，即组织TLS 证书。

# 权限管理与策略 — 证书体系（节点）



- Fabric中每个节点（背书节点、提交节点、排序节点）、客户端都会在相应目录中建立一套目录，用于存放相关的证书文件。该目录结构包括MSP 证书和TLS 证书两类。
  - TLS，存放TLS相关的证书和私钥
    - ca.crt：组织的根证书。
    - server.crt：验证本节点签名的证书，被组织根证书签名。
    - server.key：本节点的身份私钥，用来签名。

# 权限管理与策略 — 成员服务提供者 (MSP)

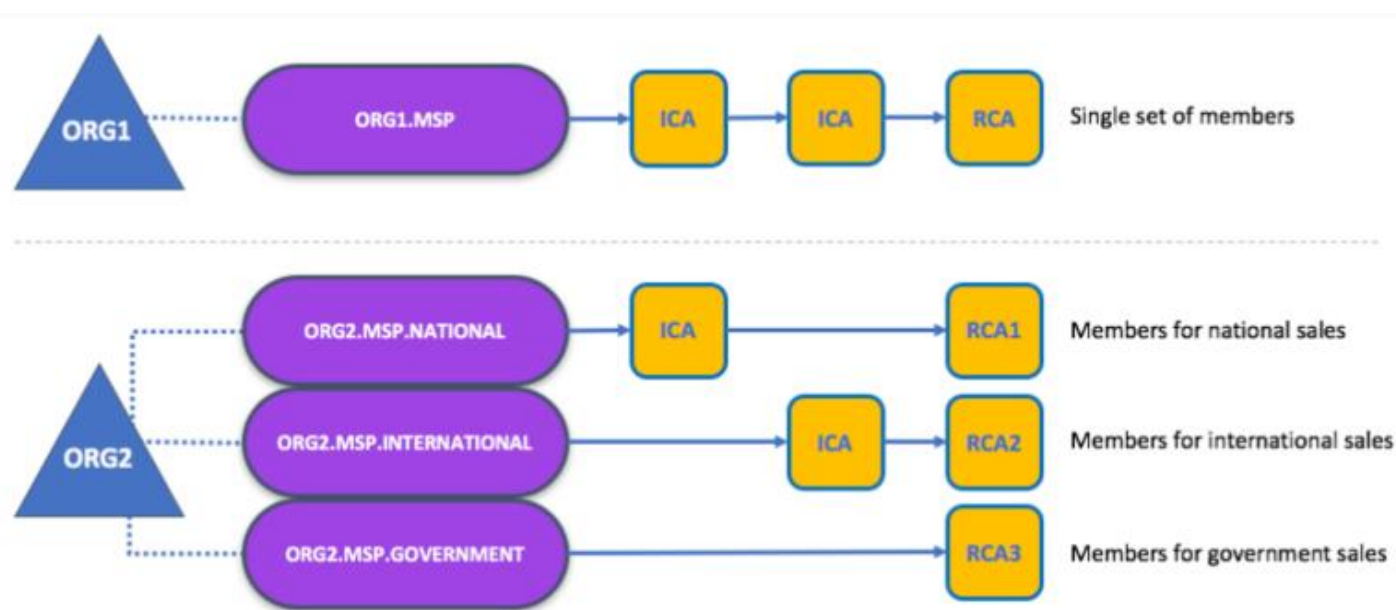


- **MSP (Membership Service Providers, 成员服务提供者)** 用于对某个资源 (组织、节点、用户等) 进行身份验证的一组机制, 是实现权限管理的基础。
  - 具体的身份;
  - 用户证书验证;
  - 用户证书撤销;
  - 签名生成和验证。
- MSP是一个可插拔模块, 其设计了成员服务相关操作的框架, 具体的证书验证与认证由相应的MSP实现模块完成, 用户可以根据要求完成个性化MSP的实现模块。

# 权限管理与策略 — MSP



- MSP的划分比较灵活，一般可以一个组织有一个MSP。
- Fabric中同一个内部的成员之间共有一个根证书，Gossip也是基于MSP进行数据同步。



# 权限管理与策略 — MSP结构



- 一个MSP一般包括9个部分，并以目录的形式存储在节点或客户端，MSP的名称是根目录，每个部分是子目录。MSP会在节点启动时把目录中的信息加载到FabricMSPConfig结构中





# 权限管理与策略 — MSP结构



- 根证书列表(**Root Cas**), 一个自签名的证书列表, MSP中的成员都是以根证书列表为根的证书, 同一根证书列表中的证书构成一个信任域;
- 中间证书列表(**Intermediate Cas**), 根证书到叶子证书的信任链中的中间部分证书, 可以按组织机构或业务等划分, 是可选参数;
- 组织单元列表(**Organizational Units (OUs)**), 组织的下级单元列表, 当基于OU限制一些权限时, 可以设置该参数, 是可选参数;
- 管理员列表(**Administrators**), 组织的管理员证书列表;



# 权限管理与策略 — MSP结构



- 吊销列表(**Revoked Certificates**), 已经从CA吊销的证书列表, 该列表中的证书没有权限访问Fabric中的资源;
- 节点身份证书(**Node Identity**), 表明节点的身份信息, 一般用于背书验证;
- 签名证书(**KeyStore for Private Key**), 节点身份证书的私钥, 用于数字签名或背书;
- **TLS根证书列表 (TLS Root Cas)**, 用于TLS通讯的根证书列表, 一般组织之间、节点与排序节点之间跨网通讯时需使用TLS进行消息传输;
- **TLS中间证书列表(TLS Intermediate CA)**, 用于TLS通讯的中间证书, 是可选参数。

# 权限管理与策略 — MSP层级

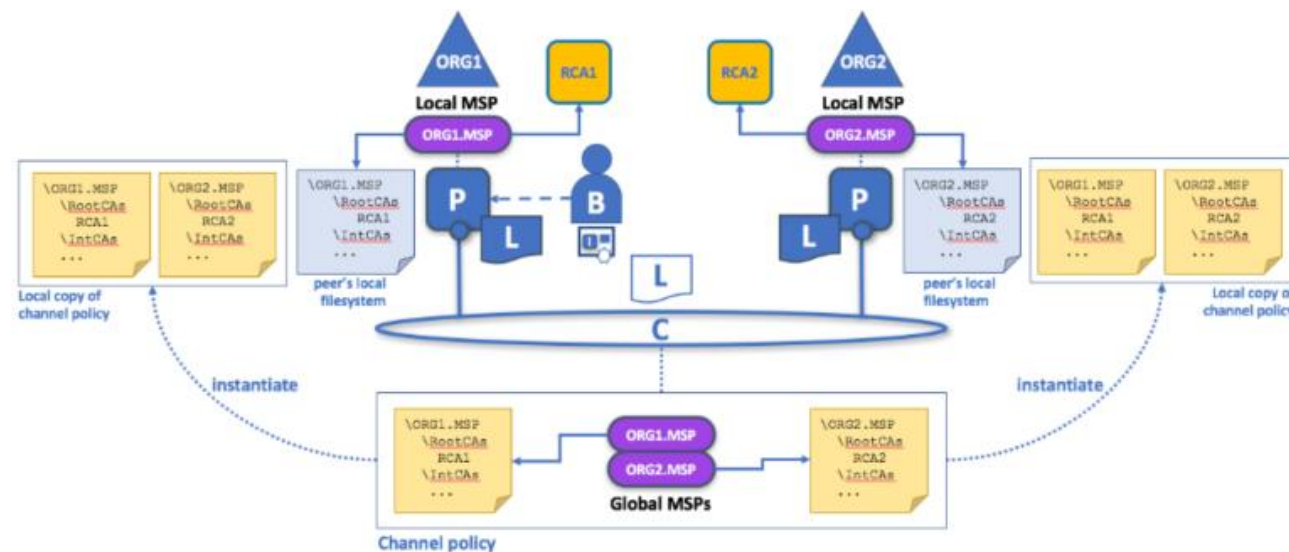


- 全局MSP(Network MSP), 面向整个Fabric网络, 配置网络管理员列表, 其成员能够完成一些网络管理任务, 如创建应用通道;
- 通道MSP(Channel MSP), 通道MSP是虚拟的MSP, 存在通道配置文件中, 当通道MSP发生改变时, 需要通过配置管理交易进行通道内节点的同步, 通道内所有节点和客户端的通道MSP是一致的;
- 本地MSP(Local MSP), 定义了本地资源(节点、客户端)的管理和参与权限, 每个节点或客户端仅仅对应一个本地MSP, 并且本地MSP以目录的形式在文件系统中保存。

# 权限管理与策略 — MSP层级



- 本地MSP和通道MSP的关系
  - 对于一个有两个组织Org1和Org2的网络，每个组织有一个节点
  - 当一个管理员B连接到节点，B具有RCA1签发的证书，当B试图在节点上安装链码时，系统会在本地MSP（ORG1-MSP）中验证B是否是Org1的合法成员，验证成功则B可以安装该链码。



**谢 谢！**