成员服务提供者 (MSP)

本文档将详细说明MSP的建立并提供MSP的最佳实践。

成员服务提供者（MSP）是 Hyperledger Fabric 的一个组件，旨在提供抽象的成员操作。

具体的，MSP将分发证书、验证证书和用户授权背后的所有加密机制和协议抽象出来。MSP可以定义它们自己的身份概念，同样还可以定义管理(身份验证)和认证(签名生成和验证)这些身份的规则。

一个 Hyperledger Fabric区块链网络可以由一个或多个MSP管理。这提供了成员操作的模块化和不同成员标准和架构之间的互操作性。

此文档的剩余部分将详述MSP在Hyperledger Fabric的建立过程，然后讨论关于其使用的最佳实践。

MSP配置

为了建立一个MSP实体，每个peer和orderer需要指定其本地的配置文件(为了使peer和orderer可以进行签名)，也为了在通道上使peer、orderer和client进行身份验证和通道成员之间的签名验证(认证)。

首先，每个MSP必须指定一个名字以便该MSP在网络内被引用(例如 `msp1`，`org2`，以及 `org3.divA`)。这是一个可以表述其所代表的在通道中联盟、组织或组织部门的名称。这个名称也被称为 *MSP Identifier* 或 *MSP ID*。每个MSP的MSPID必须是唯一的。例如，如果在系统通道建立时发现两个MSP的MSPID相同，orderer的建立将失败。

在默认的MSP实现中，需指定一些参数来允许身份(证书)验证和签名验证。这些参数从这里导出：RFC5280 ,并包括:

- 一个自签名（X.509） CA 证书列表来组成信任根（*root of trust*）
- 一个X.509证书列表来代表证书验证时需要考虑的中间证书，这些证书应该由某一个信任根颁发；中间证书是可选的参数
- 一个X.509证书列表，并拥有从某一信任根起可验证的 CA 证书路径，来代表该MSP的管理员证书；拥有管理员证书则代表拥有申请改变该MSP配置的权力(例如，根CA、中间CA)
- 一个组织单位列表，此列表应出现在该MSP的有效成员的X.509证书中；这是一个可选的配置参数，举例来说，可用于多组织使用相同信任根和中间CA，并给其成员预留OU信息
- 一个证书撤销列表(CRLs)，其中每一个对应一个列出的(根或中间)MSP CA；这是一个可选参数
- 一个自签(X.509)证书列表，用来组成TLS证书的信任根(*TLS root of trust*)
- 一个X.509证书列表来代表证书验证时需要考虑的TLS中间证书，这些证书应该由某一个TLS信任根颁发；TLS中间证书是可选的参数

该MSP的 *有效的* 身份需满足如下条件:

- 它们以X.509证书的形式存在，并拥有从某一信任根起可验证的证书路径；
- 它们不在任何证书撤销列表(CRL)中；
- 它们在其X.509证书结构的 `OU` 域中 *列举* MSP配置中的一个或多个组织单位(OU)

更多关于当前MSP实现中身份认证的信息，我们建议读者阅读文档 MSP Identity Validity Rules

除了认证相关的参数以外，为了使 MSP 启用对其进行实例化的节点进行签名或身份验证，需指定：

- 用于节点签名的签名密钥(当前只支持 ECDSA 密钥)
- 节点的 X.509 证书，这是在 MSP 的验证参数下一个有效的标识

值得注意的是 MSP 身份不会过期；它们只能被撤销（添加进证书撤销列表 CRL）。此外，目前没有支持 TLS 证书的撤销。

如何生成MSP证书以及它们的签名密钥?

为了生成MSP配置所需的X.509证书，可以使用`Openssl <https://www.openssl.org/>`_。需要强调的是Hyperledger Fabric不支持包含RSA密钥的证书。

另外也可以用 `cryptogen` 工具，它相关的操作请查看文档 入门

Hyperledger Fabric CA 也可以用来生成配置MSP的证书和密钥。

在Peer和Orderer端建立MSP

为了建立(Peer或Orderer的)本地MSP，管理员应当建立目录(例如，`$MY_PATH/mspconfig`)，其中包含一个文件和八个子目录：

1. 一个 `admincerts` 目录，其中包含PEM文件，每个PEM文件对应一个管理员证书
2. 一个 `cacerts` 目录，其中包含PEM文件，每个PEM文件对应一个根CA证书
3. (可选的)一个 `intermediatecerts` 目录，其中包含PEM文件，每个PEM文件对应一个中间CA证书
4. (可选的)一个文件 `config.yaml`，用来配置所支持的组织单位(OU)和身份分类(参见下面对应的部分)
5. (可选的)一个 `crls` 目录，包含证书撤销列表(CRLs)
6. 一个 `keystore` 目录，包含一个PEM文件，代表该节点的签名密钥，我们强调当前不支持RSA的密钥形式
7. 一个 `signcerts` 目录，包含一个PEM文件，代表该节点的X.509证书
8. (可选的)一个 `tlscacerts` 目录，其中包含PEM文件，每个PEM文件对应一个TLS根CA证书
9. (可选的)一个 `tlsintermediatecerts` 目录，其中包含PEM文件，每个PEM文件对应一个TLS中间CA证书

在节点的配置文件(对Peer来说是core.yaml，对Orderer来说是orderer.yaml)中，必须指定 mspconfig 目录的路径和节点 MSP 的 MSPID。mspconfig 目录的路径应该是环境变量 FABRIC_CFG_PATH 的相对路径，并且是 Peer 端 `mspConfigPath` 对应的参数，或是Orderer端 `LocalMSPDir` 对应的参数。节点的 MSPID 由 Peer 端 `localMspId` 指定，或由 Orderer 端 `LocalMSPID` 指定。这些变量可以被环境变量重写，在 Peer 端使用 CORE 前缀(例如，CORE_PEER_LOCALMSPID)，在 Orderer 端使用 ORDERER 前缀（例如，ORDERER_GENERAL_LOCALMSPID）。值得一提的是，在 Orderer 建立阶段，需要生成并提供给 Orderer 系统通道的创世块。创世块中所需的 MSP配置信息将在下部分详细说明。

如果想要 *重新配置* 一个 “本地” MSP，目前只能手动操作，并且Peer或Orderer需要重启。在后续版本我们计划提供在线/动态的重新配置方式(例如，不需要中止节点，使用一个受节点管理的系统链码)。

组织单元(OU)

为了配置在该 MSP 有效用户的证书中的 OU 列表，`config.yaml` 文件需指定组织单位标识。例如：

```
OrganizationalUnitIdentifiers:
  - Certificate: "cacerts/cacert1.pem"
    OrganizationalUnitIdentifier: "commercial"
  - Certificate: "cacerts/cacert2.pem"
    OrganizationalUnitIdentifier: "administrators"
```

上面的例子声明了两个组织单位标识：**commercial** 和 **administrators**。如果MSP拥有至少其中一个组织单位标识，它才是有效的。`Certificate` 域代表拥有有效标识应具有的 CA 证书或中间 CA 证书路径。该路径是相对 MSP 根目录的，并且不能为空。

身份类型

默认的MSP实现允许组织进一步将身份分类到客户端， 管理员，peer节点和基于自身x509证书的OU的排序节点。

- 在网络上进行交易的身份应该被归类为 **client**。
- 处理管理任务，例如将peer节点加入到通道或对通道配置更新交易签名， 这样的身份应该被归类为 **admin**。
- 背书或提交交易的身份应该被归类为 **peer**。
- 属于排序节点的身份应该被归类为 **orderer**。

为了定义给定MSP的客户端、管理员、peer节点和排序节点，您需要合适地设置 `config.yaml` 文件。您可以看到一个 `config.yaml` 文件的NodeOU部分的示例 如下:

```
NodeOUs:
  Enable: true
  ClientOUIdentifier:
    Certificate: "cacerts/cacert.pem"
    OrganizationalUnitIdentifier: "client"
  AdminOUIdentifier:
    Certificate: "cacerts/cacert.pem"
    OrganizationalUnitIdentifier: "admin"
  PeerOUIdentifier:
    Certificate: "cacerts/cacert.pem"
    OrganizationalUnitIdentifier: "peer"
  OrdererOUIdentifier:
    Certificate: "cacerts/cacert.pem"
    OrganizationalUnitIdentifier: "orderer"

NodeOUs:
  Enable: true
  # For each identity classification that you would like to utilize, specify
  # an OU identifier.
  # You can optionally configure that the OU identifier must be issued by a specific CA
  # or intermediate certificate from your organization. However, it is typical to NOT
  # configure a specific Certificate. By not configuring a specific Certificate, you will be
  # able to add other CA or intermediate certs later, without having to reissue all credentials.
  # For this reason, the sample below comments out the Certificate field.
  ClientOUIdentifier:
    # Certificate: "cacerts/cacert.pem"
```

```
    OrganizationalUnitIdentifier: "client"
  AdminOUIdentifier:
    # Certificate: "cacerts/cacert.pem"
    OrganizationalUnitIdentifier: "admin"
  PeerOUIdentifier:
    # Certificate: "cacerts/cacert.pem"
    OrganizationalUnitIdentifier: "peer"
  OrdererOUIdentifier:
    # Certificate: "cacerts/cacert.pem"
    OrganizationalUnitIdentifier: "orderer"
```

身份分类在 `NodeOUs.Enable` 设置为 `true` 时启用。然后通过设置key值 `NodeOUs.ClientOUIdentifier` (`NodeOUs.AdminOUIdentifier` , `NodeOUs.PeerOUIdentifier` ,``NodeOUs.OrdererOUIdentifier``)的属性 定义客户端(管理员、peer节点、排序节点)组织单元身份标识:

a. `OrganizationalUnitIdentifier` : 指x509证书需要包含的用于作为客户端 (管理员、peer节点、排序节点)的OU值。如果该字段为空，则不应用身份分类。
b. `Certificate` : (可选)将此设置为CA或中间CA证书的路径，客户端(peer节点、管理员或排序节点) 身份应在此路径下进行验证。该字段与MSP根文件夹相关。只能指定一个证书。 如果您不设置此字段，那么将根据组织的MSP配置中定义的任何CA验证身份， 如果您需要添加其他CA或中间证书，这在将来可能是理想的。

注意，如果 `NodeOUs.ClientOUIdentifier` 部分(`NodeOUs.AdminOUIdentifier` , `NodeOUs.PeerOUIdentifier` , `NodeOUs.OrdererOUIdentifier`)缺失，则分类不被应用。 如果 `NodeOUs.Enable` 设置为 `true` 并且没有定义分类key，则身份分类被认为是关闭的。

可以使用组织单元将身份分类为客户端、管理员、peer节点或排序节点。 这四种分类是相互排斥的。在身份可以被分类为客户端或peer节点之前，需要启用1.1通道功能。 要将身份分类为管理员或排序节点，需要启用1.4.3通道功能。

分类允许将身份分类为管理员(并执行管理员操作)，而不需要将证书存储在MSP的 `admincerts` 文件夹中。 相反， `admincerts` 文件夹可以保持为空，并且可以通过向管理员OU注册身份来创建管理员。 `admincerts` 文件夹中的证书仍将授予其持有者管理员的角色，前提是它们拥有客户端或管理员OU。

通道 MSP 设置

在系统创世阶段，需要指定出现在网络中的所有MSP的验证参数，并保存到系统通道的创世块。回顾一下，MSP 验证参数包含 MSP 标识、根证书列表、中间 CA 证书和管理员证书列表、OU 信息和证书撤销列表 CRL。在 Orderer 建立阶段，系统创世块将被提供给 Orderer，使 Orderer 可以认证通道建立请求。如果系统创世块包含有两个相同标识的 MSP，Orderer 将拒绝该创世块，从而导致网络启动失败。

对于应用通道，通道的创世块只需包含通道管理者的MSP验证信息。需强调的是，在将 peer 加入通道之前保证通道创世块(或最近的配置块)包含正确的MSP配置信息是 **应用自己的责任**

当使用configtxgen工具启动通道时，可以通过将MSP验证参数包含进 mspconfig 目录并在 `configtx.yaml` 相应部分设置其路径的方式配置通道MSP。

通道上 MSP 的 *重新配置*，MSP 管理员证书的持有者在创建 `config_update` 事务时，将声明与该 MSP 的已获得CA的证书相关的证书撤销列表。随后被管理员控制的客户端应用将在 MSP 出现的通道上声明

这次更新。

最佳实践

在这部分我们将详细说明对MSP配置的通用场景下的最佳实践

1) 组织/企业 和 MSP 之间的映射

我们建议组织和MSP之间是一对一映射的。如果要使用其他类型的映射，需考虑以下情况：

- **一个组织使用多个MSP。** 这对应的情况是一个组织有多个部门，每个MSP代表一个部门，出现这种情况可以是独立管理的原因，也可能出于隐私考虑。在这种情况下，一个peer节点只能被单一MSP拥有，并且不能将其他MSP下peer识别成同组织的peer。这意味着peer节点可以通过gossip组织域将数据分享给同部门内的其他peer节点，但不能分享给组成实际组织的全体。
- **多组织使用一个MSP。** 这对应的情况是多个组织组成联盟，每个组织都被类似的成员架构管理。要知道，不论是否属于同一实际组织，peer的组织域消息将传播给同MSP下的其他peer节点。这将限制MSP定义和(或)peer配置的粒度。

2) 一个组织有不同分部(组织单元)，想要授予不同通道访问权

两种处理方法：

- **定义一个可以容纳所有组织成员的MSP**。该MSP的配置将由根CA、中间CA和管理员证书列表；以及成员标识包括成员所属的组织单元(`OU`)组成。随后定义策略来捕获某一特定 `OU` 的成员，这些策略将组成通道的读/写策略或链码的背书策略。这种方法的局限性是gossip peer节点将把拥有和其相同成员标识的peer当成同组织成员，并因此与它们传播组织域信息(例如状态信息)。
- **给每一个分部定义一个MSP**。这涉及到给每个分部指定一组证书，包含根CA证书、中间CA证书和管理员证书，这样能够做到MSP之间没有重复的证书路径。这意味着，每个分部采用不同的中间CA。这么做的缺点是需要管理多个MSP，但是确实绕开了上面方法出现的问题。我们也可以使用MSP配置里的OU扩展项来实现对每个分部定义一个MSP。

3) 区分同一组织下的client和peer

在很多情况下，会要求一个身份的 “type” 是可以被检索的(例如，可能有需求要求背书必须由peer节点提供，不能是client或单独的orderer节点)。

对这种要求的支持是有限的。

实现这种区分的一种方式是为每种节点类型创建单独的中间CA，一个给client，一个给peer或orderer，并分别配置两个不同的MSP。组织加入到的通道需要同时包含两个MSP，但背书策略只部署在peer的MSP。这将最终导致组织被映射到两个MSP实例，并且对peer和client的交互产生一些后果。

由于同一组织的所有peer还是属于同一个MSP，Gossip不会被严重的影响。Peer可以基于本地MSP策略来约束特定系统链码的执行。例如，peer可以只执行 “joinChannel”请求，如果这个请求是被一个只能是client的本地MSP的管理员签名的(终端用户应该是请求的起点)。我们可以绕过这个矛盾，只要我们接受该MSP的管理员是该peer/orderer的唯一client。

这种方法要考虑的另一个点是peer基于请求发起者本地MSP的资格来授权事件注册请求。很明显，由于请求发起者是一个client，它经常被当作是属于与该peer不同的MSP，因此peer将拒绝请求。

4) 管理员和CA证书

将MSP管理员证书设成与该MSP的 `root of trust` 或中间CA的证书不同非常重要。将管理成员组件和分发新和(或)验证证书的职责分开是常规(安全的)做法。

5) 将一个中间CA列入黑名单

前面提到，可以通过重新配置机制(对本地MSP实例手动重新配置，并对通道的MSP适当的构建 `config_update` 消息)对MSP进行重新配置。很明显，有两种方式将一个中间CA列入黑名单：

1. 重新配置MSP，使其中间CA证书列表不再包含该中间CA。对本地已配置的MSP来说， 这意味着这个CA的证书将从 `intermediatecerts` 目录移除。
2. 重新配置MSP,使其包含一个由信任根颁发的证书撤销列表，该列表包含提到的中间CA的证书。

当前的 MSP 实现中，我们只支持方式(1)，因为其更简单，并且不要求将不再考虑的中间 CA 列入黑名单。

6) CA 和 TLS CA

MSP身份的根CA和MSP TLS根CA(以及相关的中间CA)需要在不同的目录被定义。这是为了避免不同类证书之间产生混淆。虽然没有禁止MSP身份和TLS证书使用相同的CA，但这里建议避免在生成环境这样做。