滴雨科技 Fabric CA 证书用户指南

Hyperledger Fabric CA 是 Hyperledger Fabric 的证书颁发机构（CA）。

它提供了以下功能：

- 身份注册，或连接到 LDAP 作为用户注册表
- 颁发入学证书（ECerts）
- 证书续签和吊销

Hyperledger Fabric CA 由服务器和客户端组件组成，如本文档后面所述。

对于有兴趣向 Hyperledger Fabric CA 做出贡献的开发人员，请参阅 Fabric CA 存储库以获取更多信息。

1. 目录

1. 总览

下图说明了 Hyperledger Fabric CA 服务器如何适合整个 Hyperledger Fabric 体系结构。

与 Hyperledger Fabric CA 服务器进行交互的方式有两种：通过 Hyperledger Fabric CA 客户端或通过 Fabric SDK 之一。与 Hyperledger Fabric CA 服务器的所有通信都是通过 REST API。请参阅 *fabric-ca / swagger / swagger-fabric-ca.json* 以获取这些 REST API 的详细文档。您可以通过 http://editor2.swagger.io 在线编辑器查看此文档。

Hyperledger Fabric CA 客户端或 SDK 可以连接到 Hyperledger Fabric CA 服务器群集中的服务器。在图的右上部分对此进行了说明。客户端路由到 HA 代理终结点，该终结点将负载平衡流量到结构 CA 服务器群集成员之一。

群集中的所有 Hyperledger Fabric CA 服务器共享同一数据库，以跟踪身份和证书。如果配置了 LDAP，则身份信息将保留在 LDAP 中而不是数据库中。

一个服务器可能包含多个 CA。每个 CA 可以是根 CA 或中间 CA。每个中间 CA 都有一个父 CA，它可以是根 CA 或另一个中间 CA。

2. 入门

1 先决条件

- 去 1.10+安装
- GOPATH 环境变量设置正确
- 已安装 libtool 和 libtdhl-dev 软件包

以下在 Ubuntu 上安装 libtool 依赖项：

```
执行 sudo  易于  安装  的 libtool  的 libltdl - 开发
```

以下内容在 MacOSX 上安装了 libtool 依赖项：

```
brew  安装  libtool
```

注意

如果通过 Homebrew 安装 libtool，则在 MacOSX 上不需要 libtldl-dev

有关 libtool 的更多信息，请参见 https://www.gnu.org/software/libtool。

有关 libltdl-dev 的更多信息，请参见 https://www.gnu.org/software/libtool/manual/html_node/Using-libltdl.html。

2 安装

以下内容将$ capath / bin 中的 *fabric-ca-server* 和 *fabric-ca-client* 二进制文件安装。

```
go get -u github.com/hyperledger/fabric-ca/cmd/...
```

注意：如果您已经克隆了 fabric-ca 存储库，请在运行上面的“ go get”命令之前确保您位于 master 分支上。否则，您可能会看到以下错误：

```
<gopath>/src/github.com/hyperledger/fabric-ca; git pull --ff-only
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=<remote>/<branch> tlsdoc

package github.com/hyperledger/fabric-ca/cmd/fabric-ca-client: exit status 1
```

3 本地启动服务器

以下将使用默认设置启动 *fabric-ca-server*。

```
fabric-ca-server start -b admin:adminpw
```

该 *-b* 选项提供了一个引导管理员报名 ID 和秘密; 如果未使用“ ldap.enabled”设置启用 LDAP，则这是必需的。

将 在本地目录中创建一个名为 *fabric-ca-server-config.yaml* 的默认配置文件，可以对其进行自定义。

4 通过 Docker 启动服务器

> **Docker 中心**

前往：https：//hub.docker.com/r/hyperledger/fabric-ca/tags/

查找与要提取的 fabric-ca 的体系结构和版本匹配的标签。

导航至 *$ GOPATH / src / github.com / hyperledger / fabric-ca / docker / server* 并在编辑器中打开 docker-compose.yml。

更改*图像*线以反映先前找到的标签。对于 beta 版的 x86 体系结构，该文件可能看起来像这样。

```
fabric-ca-server:
  image: hyperledger/fabric-ca:x86_64-1.0.0-beta
  container_name: fabric-ca-server
  ports:
    - "7054:7054"
  environment:
    - FABRIC_CA_HOME=/etc/hyperledger/fabric-ca-server
  volumes:
    - "./fabric-ca-server:/etc/hyperledger/fabric-ca-server"
  command: sh -c 'fabric-ca-server start -b admin:adminpw'
```

在与 docker-compose.yml 文件相同的目录中打开一个终端，然后执行以下命令：

```
# docker-compose up -d
```

如果指定的 fabric-ca 映像尚不存在，这将在 Compose 文件中将其下拉，并启动 Fabric-ca 服务器的实例。

构建自己的 Docker 映像

您可以通过 Docker Compose 构建和启动服务器，如下所示。

```
cd $GOPATH/src/github.com/hyperledger/fabric-ca
make docker
cd docker/server
docker-compose up -d
```

hyperledger / fabric-ca Docker 映像包含 fabric-ca-server 和 fabric-ca-client。

```
# cd $GOPATH/src/github.com/hyperledger/fabric-ca
# FABRIC_CA_DYNAMIC_LINK=true make docker
# cd docker/server
# docker-compose up -d
```

5 探索 Fabric CA CLI

为方便起见，本节仅提供 Fabric CA 服务器和客户端的使用情况消息。以下各节提供了其他用法信息。

以下链接显示了服务器命令行和 客户端命令行。

注意

请注意，可以通过使用逗号分隔的列表元素指定选项，或者通过多次指定选项来指定作为字符串切片（列表）的命令行选项，每个选项都使用组成列表的字符串值。例如，要指定 host1 和 host2 作为 csr.hosts 选项，可以传递或 。使用前一种格式时，请确保在任何逗号前后没有空格。 --csr.hosts 'host1,host2'--csr.hosts host1 --csr.hosts host2

6 配置设定

Fabric CA 提供了 3 种在 Fabric CA 服务器和客户端上配置设置的方式。优先顺序为：

1. CLI 标志
2. 环境变量
3. 配置文件

在本文档的其余部分中，我们指的是对配置文件进行更改。但是，可以通过环境变量或 CLI 标志来覆盖配置文件更改。

例如，如果客户端配置文件中包含以下内容：

```
tls:
  # Enable TLS (default: false)
  enabled: false

  # TLS for the client's listenting port (default: false)
  certfiles:
  client:
    certfile: cert.pem
    keyfile:
```

以下环境变量可用于覆盖 `cert.pem` 配置文件中的设置：

```
export FABRIC_CA_CLIENT_TLS_CLIENT_CERTFILE=cert2.pem
```

如果我们想覆盖环境变量和配置文件，则可以使用命令行标志。

```
fabric-ca-client enroll --tls.client.certfile cert3.pem
```

相同的方法适用于 fabric-ca-server，除了使用而不是 `FABIRC_CA_CLIENT` 用作环境变量的前缀 之外 `FABRIC_CA_SERVER`。

> **在文件路径上应注意的**

Fabric CA 服务器和客户端配置文件中所有指定文件名的属性都支持相对路径和绝对路径。相对路径相对于配置文件所在的 config 目录。例如，如果配置目录是 `~/config` 和所述 TLS 部，如下所示，该织物 CA 服务器或客户端将寻找 `root.pem` 在文件 `~/config` 目录中，`cert.pem` 文件中的 `~/config/certs` 目录和 `key.pem` 文件中的 `/abs/path` 目录

```
tls:
  enabled: true
  certfiles:
    - root.pem
  client:
    certfile: certs/cert.pem
    keyfile: /abs/path/key.pem
```

3. Fabric CA 服务器

本节介绍 Fabric CA 服务器。

您可以在启动 Fabric CA 服务器之前对其进行初始化。这为您提供了生成默认配置文件的机会，可以在启动服务器之前对其进行检查和自定义。

Fabric CA 服务器的主目录确定如下：

- 如果设置了–home 命令行选项，请使用其值
- 否则，如果 `FABRIC_CA_SERVER_HOME` 设置了环境变量，请使用其值
- 否则，如果 `FABRIC_CA_HOME` 设置了环境变量，请使用其值
- 否则，如果 `CA_CFG_PATH` 设置了环境变量，请使用其值
- 否则，使用当前工作目录

对于此服务器部分的其余部分，我们假设您已将 `FABRIC_CA_HOME` 环境变量设置为 `$HOME/fabric-ca/server`。

以下说明假定服务器配置文件位于服务器的主目录中。

1 初始化服务器

初始化 Fabric CA 服务器，如下所示：

```
fabric-ca-server init -b admin:adminpw
```

该 `-b`（引导标识）选项是必需的初始化时，LDAP 被禁用。要启动 Fabric CA 服务器，至少需要一个引导程序标识。此身份是服务器管理员。

服务器配置文件包含可以配置的证书签名请求（CSR）部分。以下是示例 CSR。

```
cn: fabric-ca-server
names:
   - C: US
     ST: "North Carolina"
     L:
     O: Hyperledger
     OU: Fabric
hosts:
  - host1.example.com
  - localhost
ca:
   expiry: 131400h
   pathlength: 1
```

上面的所有字段都与 X.509 签名密钥和证书有关，这些签名和证书由生成。这对应于服务器配置文件中的 和 文件。字段如下：`fabric-ca-server initca.certfileca.keyfile`

- **cn** 是通用名称
- **O** 是组织名称
- **OU** 是组织单位
- **L** 是位置或城市
- **ST** 是状态
- **C** 是国家

如果需要 CSR 的自定义值，则可以自定义配置文件，删除 `ca.certfile` 和 `ca.keyfile` 配置项指定的文件，然后 再次运行命令。`fabric-ca-server init -b admin:adminpw`

除非指定该选项，否则该命令将生成自签名的 CA 证书。如果指定，则服务器的 CA 证书由父 Fabric CA 服务器签名。为了向父 Fabric CA 服务器进行身份验证，URL 的格式必须为，其中<enrollmentID>和<secret>对应于具有 “ hf.IntermediateCA” 属性且其值等于 “ true” 的身份。该命令还会在服务器的主目录中生成一个名为 **fabric-ca-server-config.yaml** 的默认配置文件。`fabric-ca-server init-u <parent-fabric-ca-server-URL>-u<scheme>://<enrollmentID>:<secret>@<host>:<port>fabric-ca-server init`

如果你想在面料 CA 服务器使用 CA 签名证书和密钥文件，你提供，你必须把你的文件中被引用的位置 `ca.certfile`，并 `ca.keyfile` 分别。两个文件都必须经过 PEM 编码，并且不能加密。更具体地说，CA 证书文件 的内容必须以开头，密钥文件的内容必须以而不 开头。`-----BEGIN CERTIFICATE----------BEGIN PRIVATE KEY----------BEGIN ENCRYPTED PRIVATE KEY-----`

算法和密钥大小

可以自定义 CSR 以生成支持椭圆曲线（ECDSA）的 X.509 证书和密钥。以下设置是使用曲线 `prime256v1` 和签名算法 实现椭圆曲线数字签名算法（ECDSA）的示例 `ecdsa-with-SHA256`：

```
key:
   algo: ecdsa
   size: 256
```

算法和密钥大小的选择取决于安全性需求。

椭圆曲线（ECDSA）提供以下密钥大小选项：

尺寸	ASN1 OID	签名算法
256	prime256v1	带 SHA256 的 ecdsa
384	secp384r1	带 SHA384 的 ecdsa
521	secp521r1	带 SHA512 的 ecdsa

2 启动服务器

启动 Fabric CA 服务器，如下所示：

```
fabric-ca-server start -b <admin>:<adminpw>
```

如果服务器之前未进行过初始化，它将在首次启动时对其进行初始化。在初始化期间，服务器将生成 ca-cert.pem 和 ca-key.pem 文件（如果尚不存在），并且还将创建默认配置文件（如果不存在）。请参阅初始化 Fabric CA 服务器部分。

除非将 Fabric CA 服务器配置为使用 LDAP，否则必须为它配置至少一个预注册的引导程序身份，以使您能够注册和注册其他身份。该 `-b` 选项指定引导程序标识的名称和密码。

要使 Fabric CA 服务器侦听 `https` 而不是 侦听 `http`，请设置 `tls.enabled` 为 `true`。

安全警告：Fabric CA 服务器应始终在启用 TLS（`tls.enabled` 设置为 true）的情况下启动。否则，服务器容易受到攻击者访问网络流量的攻击。

要限制可以使用同一密码（或密码）进行注册的次数，请将 `registry.maxenrollments` 配置文件中的设置为适当的值。如果将值设置为 1，Fabric CA 服务器将仅将一次密码用于特定的注册 ID。如果将该值设置为-1，则 Fabric CA 服务器对可将机密重新用于注册的次数没有限制。默认值为-1。将该值设置为 0，Fabric CA 服务器将禁用所有身份的注册，并且将不允许身份注册。

现在，Fabric CA 服务器应该正在侦听端口 7054。

如果不想将 Fabric CA 服务器配置为在群集中运行或使用 LDAP ，则可以跳至 Fabric CA Client 部分。

3 配置数据库

本节介绍如何配置 Fabric CA 服务器以连接到 PostgreSQL 或 MySQL 数据库。默认数据库为 SQLite，默认数据库文件 `fabric-ca-server.db` 位于 Fabric CA 服务器的主目录中。

如果您不关心在群集中运行 Fabric CA 服务器，则可以跳过本节；请参阅第 14 页的『创建 CA。否则，您必须按照以下说明配置 PostgreSQL 或 MySQL。Fabric CA 在群集设置中支持以下数据库版本：

- PostgreSQL：9.5.5 或更高版本
- MySQL：5.7 或更高版本

PostgreSQL 的

可以将以下示例添加到服务器的配置文件中，以便连接到 PostgreSQL 数据库。确保适当地自定义各种值。数据库名称中允许使用哪些字符有限制。请参考以下 Postgres 文档以获取更多信息：https：//www.postgresql.org/docs/current/static/sql-syntax-lexical.html#SQL-SYNTAX-IDENTIFIERS

```
db:
  type: postgres
  datasource: host=localhost port=5432 user=Username password=Password dbname=fabric_ca
sslmode=verify-full
```

指定 *sslmode* 可配置 SSL 身份验证的类型。sslmode 的有效值为：

Mode	Description
disable	No SSL
require	Always SSL (skip verification)
verify-ca	Always SSL (verify that the certificate presented by the server was signed by a trusted CA)
verify-full	Same as verify-ca AND verify that the certificate presented by the server was signed by a trusted CA and the server hostname matches the one in the certificate

如果要使用 TLS，则 `db.tls` 必须指定 Fabric CA 服务器配置文件中的部分。如果在 PostgreSQL 服务器上启用了 SSL 客户端身份验证，则还必须在该 `db.tls.client` 部分中指定客户端证书和密钥文件。以下是本 `db.tls` 节的示例：

```
db:
```

```
...
tls:
    enabled: true
    certfiles:
      - db-server-cert.pem
    client:
          certfile: db-client-cert.pem
          keyfile: db-client-key.pem
```

certfiles -PEM 编码的受信任根证书文件的列表。
certfile 和 **keyfile** -Fabric CA 服务器用来与 PostgreSQL 服务器安全通信的 PEM 编码的证书和密钥文件
PostgreSQL SSL 配置

在 PostgreSQL 服务器上配置 SSL 的基本说明：

1. 在 postgresql.conf 中，取消注释 SSL 并将其设置为“ on”（SSL = on）
2. 将证书和密钥文件放置在 PostgreSQL 数据目录中。

生成自签名证书的说明：https：//www.postgresql.org/docs/9.5/static/ssl-tcp.html

注意：自签名证书仅用于测试目的，不应在生产环境中使用

PostgreSQL 服务器-需要客户端证书

1. 将您信任的证书颁发机构（CA）的证书放在 PostgreSQL 数据目录中的文件 root.crt 中
2. 在 postgresql.conf 中，将“ ssl_ca_file”设置为指向客户端的根证书（CA 证书）
3. 在 pg_hba.conf 中适当的 hostssl 行上将 clientcert 参数设置为 1。

有关在 PostgreSQL 服务器上配置 SSL 的更多详细信息，请参考以下 PostgreSQL 文档：
https：//www.postgresql.org/docs/9.4/static/libpq-ssl.html

MySQL

可以将以下示例添加到 Fabric CA 服务器配置文件中，以便连接到 MySQL 数据库。确保适当地自定义各种值。数据库名称中允许使用哪些字符有限制。请参考以下 MySQL 文档以获取更多信息：
https：//dev.mysql.com/doc/refman/5.7/en/identifiers.html

在 MySQL 5.7.X 上，某些模式会影响服务器是否允许将“ 0000-00-00”作为有效日期。可能有必要放宽 MySQL 服务器使用的模式。我们希望服务器能够接受零日期值。

在 my.cnf 中，找到配置选项 *sql_mode* 并删除 *NO_ZERO_DATE（*如果存在）。进行此更改后，重新启动 MySQL 服务器。

请参考以下有关不同模式的 MySQL 文档，并为正在使用的特定 MySQL 版本选择适当的设置。

https://dev.mysql.com/doc/refman/5.7/zh-CN/sql-mode.html

```
db:
  type: mysql
  datasource: root:rootpw@tcp(localhost:3306)/fabric_ca?parseTime=true&tls=custom
```

如果通过 TLS 连接到 MySQL 服务器，`db.tls.client` 则也需要该部分，如上面 **PostgreSQL** 部分中所述。

MySQL SSL 配置

在 MySQL 服务器上配置 SSL 的基本说明：

1. 打开或创建服务器的 my.cnf 文件。在[mysqld]部分下面的行中添加或取消注释。这些应该指向服务器的密钥和证书以及根 CA 证书。

   有关创建服务器和客户端证书的说明：http：//dev.mysql.com/doc/refman/5.7/en/creating-ssl-files-using-openssl.html

   [mysqld] ssl-ca = ca-cert.pem ssl-cert = server-cert.pem ssl-key = server-key.pem

   可以运行以下查询以确认已启用 SSL。

   mysql>显示全局变量，如'have_%ssl';

   应该看到：

变量名	值
have_openssl	是
have_ssl	是

2. 服务器端 SSL 配置完成后，下一步是创建一个有权通过 SSL 访问 MySQL 服务器的用户。为此，登录到 MySQL 服务器，然后键入：

   mysql>授予所有特权。要通过“密码”标识为“ ssluser” @“%”，需要 SSL；mysql>冲洗特权；

   如果要提供用户访问服务器的特定 IP 地址，请将“%”更改为特定 IP 地址。

MySQL 服务器-需要客户端证书

安全连接的选项与服务器端使用的选项相似。

- ssl-ca 标识证书颁发机构（CA）证书。如果使用此选项，则必须指定服务器使用的相同证书。
- ssl-cert 标识 MySQL 服务器的证书。
- ssl-key 标识 MySQL 服务器的私钥。

假设您要使用没有特殊加密要求的帐户或使用包含 REQUIRE SSL 选项的 GRANT 语句创建的帐户进行连接。作为推荐的一组安全连接选项，至少使用–ssl-cert 和–ssl-key 选项启动 MySQL 服务器。然后 `db.tls.certfiles` 在服务器配置文件中设置属性，然后启动 Fabric CA 服务器。

要要求还指定客户端证书，请使用 REQUIRE X509 选项创建帐户。然后，客户端还必须指定正确的客户端密钥和证书文件；否则，MySQL 服务器将拒绝连接。要为 Fabric CA 服务器指定客户端密钥和证书文件，请设置 `db.tls.client.certfile` 和 `db.tls.client.keyfile` 配置属性。

4 配置 LDAP

可以将 Fabric CA 服务器配置为从 LDAP 服务器读取。

特别是，Fabric CA 服务器可以连接到 LDAP 服务器以执行以下操作：

- 在注册之前验证身份
- 检索用于授权的身份的属性值。

修改 Fabric CA 服务器的配置文件的 LDAP 部分，以配置服务器以连接到 LDAP 服务器。

```
ldap:
   # Enables or disables the LDAP client (default: false)
   enabled: false
   # The URL of the LDAP server
   url: <scheme>://<adminDN>:<adminPassword>@<host>:<port>/<base>
   userfilter: <filter>
   attribute:
      # 'names' is an array of strings that identify the specific attributes
      # which are requested from the LDAP server.
      names: <LDAPAttrs>
      # The 'converters' section is used to convert LDAP attribute values
      # to fabric CA attribute values.
      #
      # For example, the following converts an LDAP 'uid' attribute
      # whose value begins with 'revoker' to a fabric CA attribute
      # named "hf.Revoker" with a value of "true" (because the expression
      # evaluates to true).
      #    converters:
      #       - name: hf.Revoker
      #         value: attr("uid") =~ "revoker*"
      #
      # As another example, assume a user has an LDAP attribute named
      # 'member' which has multiple values of "dn1", "dn2", and "dn3".
      # Further assume the following configuration.
      #    converters:
      #       - name: myAttr
      #         value: map(attr("member"),"groups")
      #    maps:
      #       groups:
      #          - name: dn1
      #            value: client
      #          - name: dn2
      #            value: peer
      # The value of the user's 'myAttr' attribute is then computed to be
```

```
    # "client,peer,dn3".  This is because the value of 'attr("member")' is
    # "dn1,dn2,dn3", and the call to 'map' with a 2nd argument of
    # "group" replaces "dn1" with "client" and "dn2" with "peer".
    converters:
      - name: <fcaAttrName>
        value: <fcaExpr>
    maps:
      <mapName>:
          - name: <from>
            value: <to>
```

在这里：

- scheme 是一个 *LDAP* 或 *LDAPS*；
- adminDN 是管理员用户的专有名称；
- pass 是管理员用户的密码；
- host 是 LDAP 服务器的主机名或 IP 地址；
- port 是可选端口号，其中 *ldap* 缺省为 389 ， *ldaps* 缺省为 636 ；
- base 是用于搜索的 LDAP 树的可选根；
- filter 是在搜索以将登录用户名转换为专有名称时使用的过滤器。例如，一个 (uid=%s) 搜索值使用 uid 属性值（其值为登录用户名）搜索 LDAP 条目。同样， (email=%s) 可用于登录电子邮件地址。
- LDAPAttrs 是代表用户从 LDAP 服务器请求的 LDAP 属性名称的数组；
- attribute.converters 部分用于将 LDAP 属性转换为结构 CA 属性，其中* fcaAttrName 是结构 CA 属性的名称；* fcaExpr 是一个表达式，其评估值分配给结构 CA 属性。例如，假设<LDAPAttrs>是[“ uid”]，<fcaAttrName>是'hf.Revoker'，而<fcaExpr>是'attr （“ uid”） =∽“ revoker *”'。这意味着代表用户从 LDAP 服务器请求一个名为“ uid”的属性。如果用户的'uid'LDAP 属性的值以'revoker'开头，则为用户提供'hf.Revoker'属性的'true'值。否则，为用户提供'hf.Revoker'属性的'false'值。
- attribute.maps 部分用于映射 LDAP 响应值。典型的用例是将与 LDAP 组关联的专有名称映射到身份类型。

LDAP 表达式语言使用 govaluate 软件包，
如 https://github.com/Knetic/govaluate/blob/master/MANUAL.md 中所述。这定义了运算符（例如“ =∽”）和文字（例如“ revoker *”），这是一个正则表达式。扩展基本 govaluate 语言的 LDAP 特定变量和函数如下：

- DN 是一个等于用户专有名称的变量。
- affiliation 是一个等于用户所属关系的变量。
- attr 是带有 1 个或 2 个参数的函数。第一个参数是 LDAP 属性名称。第二个参数是分隔符字符串，用于将多个值连接到单个字符串中；默认的分隔符字符串是“，”。该 attr 函数始终返回“字符串”类型的值。

- `map` 是一个带有 2 个参数的函数。第一个参数是任何字符串。第二个参数是映射的名称，该映射用于对第一个参数中的字符串执行字符串替换。
- `if` 是一个带有 3 个参数的函数，其中第一个参数必须解析为布尔值。如果计算结果为 true，则返回第二个参数；否则，返回 0。否则，返回第三个参数。

例如，如果用户具有以“ O = org1，C = US”结尾的专有名称，或者用户具有以“ org1.dept2”开头的从属关系，则以下表达式的值为 true。属性为“ true”。

DN =~ “*O=org1,C=US” || (affiliation =~ “org1.dept2.*” && attr(‘admin’) = ‘true’)

注意：由于该 `attr` 函数始终返回类型为'string'的值，因此可能不使用数字运算符来构造表达式。例如，以下不是有效的表达式：

```
value: attr("gidNumber) >= 10000 && attr("gidNumber) < 10006
```

或者，可以使用如下所示的用引号引起来的正则表达式来返回等效结果：

```
value: attr("gidNumber") =~ "1000[0-5]$" || attr("mail") == "root@example.com"
```

以下是 Docker 映像位于的 OpenLDAP 服务器的默认设置的示例配置部分 `https://github.com/osixia/docker-openldap`。

```
ldap:
   enabled: true
   url: ldap://cn=admin,dc=example,dc=org:admin@localhost:10389/dc=example,dc=org
   userfilter: (uid=%s)
```

在参考资料中 `FABRIC_CA/scripts/run-ldap-tests` 查找用于启动 OpenLDAP Docker 映像，对其进行配置，在中运行 LDAP 测试 `FABRIC_CA/cli/server/ldap/ldap_test.go` 并停止 OpenLDAP 服务器的脚本 。

配置 LDAP 后，注册的工作方式如下：

- Fabric CA 客户端或客户端 SDK 发送带有基本授权标头的注册请求。
- Fabric CA 服务器接收注册请求，在授权标头中解码身份名称和密码，使用配置文件中的“ userfilter”查找与身份名称关联的 DN（专有名称），然后尝试使用 LDAP 绑定身份的密码。如果 LDAP 绑定成功，则注册过程将被授权并可以继续进行。

5 设置集群

您可以使用任何 IP Sprayer 将负载均衡到 Fabric CA 服务器群集。本节提供了有关如何设置 Haproxy 以便路由到 Fabric CA 服务器群集的示例。确保更改主机名和端口以反映您的 Fabric CA 服务器的设置。

haproxy.conf

```
global
      maxconn 4096
```

```
      daemon

defaults
      mode http
      maxconn 2000
      timeout connect 5000
      timeout client 50000
      timeout server 50000

listen http-in
      bind *:7054
      balance roundrobin
      server server1 hostname1:port
      server server2 hostname2:port
      server server3 hostname3:port
```

注意：如果使用 TLS，则需要使用。`mode tcp`

6 设置多个 CA

默认情况下，fabric-ca 服务器由一个默认的 CA 组成。但是，可以通过使用 *cafiles* 或 *cacount* 配置选项将其他 CA 添加到单个服务器。每个其他 CA 将有其自己的主目录。

> **CA 帐户：**

该 *cacount* 提供了一种快速启动的默认额外的 CA X 号。主目录将相对于服务器目录。使用此选项，目录结构将如下所示：

```
--<Server Home>
  |--ca
    |--ca1
    |--ca2
```

每个附加的 CA 将在其主目录中生成一个默认配置文件，该配置文件中将包含一个唯一的 CA 名称。

例如，以下命令将启动 2 个默认 CA 实例：

```
fabric-ca-server start -b admin:adminpw --cacount 2
```

> **CA 文件 cafiles：**

如果在使用 cafiles 配置选项时未提供绝对路径，则 CA 主目录将相对于服务器目录。

要使用此选项，必须已经为要启动的每个 CA 生成并配置了 CA 配置文件。每个配置文件必须具有唯一的 CA 名称和公用名（CN），否则服务器将无法启动，因为这些名称必须是唯一的。CA 配置文件将覆盖任何默认的 CA 配置，并且 CA 配置文件中所有丢失的选项都将被默认 CA 的值替换。

优先顺序如下：

1. CA 配置文件
2. 默认 CA CLI 标志

3. 默认 CA 环境变量
4. 默认 CA 配置文件

CA 配置文件必须至少包含以下内容：

```
ca:
# Name of this CA
name: <CANAME>

csr:
  cn: <COMMONNAME>
```

您可以按以下方式配置目录结构：

```
--<Server Home>
  |--ca
    |--ca1
      |-- fabric-ca-config.yaml
    |--ca2
      |-- fabric-ca-config.yaml
```

例如，以下命令将启动两个自定义的 CA 实例：

```
fabric-ca-server start -b admin:adminpw --cafiles ca/ca1/fabric-ca-config.yaml
--cafiles ca/ca2/fabric-ca-config.yaml
```

7 注册中间 CA

为了为中间 CA 创建 CA 签名证书，中间 CA 必须以与父 CA 相同的方式向父 CA 注册。这是通过使用-u 选项指定父 CA 的 URL 以及注册 ID 和密码来完成的，如下所示。与该注册 ID 关联的身份必须具有名称为“hf.IntermediateCA”和值为“true”的属性。颁发证书的 CN（或通用名称）将设置为注册 ID。如果中间 CA 尝试显式指定 CN 值，则会发生错误。

```
fabric-ca-server start -b admin:adminpw -u
http://<enrollmentID>:<secret>@<parentserver>:<parentport>
```

有关其他中间 CA 标志，请参阅 Fabric CA 服务器的配置文件格式部分。

8 升级服务器

在升级 Fabric CA 客户端之前，必须先升级 Fabric CA 服务器。升级之前，建议备份当前数据库：

- 如果使用 sqlite3，请备份当前数据库文件（默认情况下命名为 fabric-ca-server.db）。
- 对于其他数据库类型，请使用适当的备份/复制机制。

要升级 Fabric CA 服务器的单个实例：

1. 停止 fabric-ca-server 进程。
2. 确保当前数据库已备份。
3. 用升级版本替换以前的 fabric-ca-server 二进制文件。

4. 启动 fabric-ca-server 进程。
5. 使用以下命令验证 fabric-ca-server 进程是否可用，其中<host>是启动服务器的主机名：

```
fabric-ca-client getcainfo -u http://<host>:7054
```

升级集群：

要使用 MySQL 或 Postgres 数据库升级 Fabric-ca-server 实例集群，请执行以下过程。我们假设您正在使用 haproxy 分别对主机 1 和 host2 上的两个 Fabric-ca-server 群集成员进行负载平衡，并且都侦听端口 7054。完成此过程之后，您将对已升级的 Fabric-ca-server 群集成员进行负载平衡。分别在 host3 和 host4 上侦听端口 7054。

为了使用 haproxy 统计信息监视更改，请启用统计信息收集。将以下行添加到 haproxy 配置文件的 global 部分：

```
stats socket /var/run/haproxy.sock mode 666 level operator
stats timeout 2m
```

重新启动 haproxy 以获取更改：

```
# haproxy -f <配置文件> -st $（pgrep haproxy）
```

要显示来自 haproxy“ show stat”命令的摘要信息，以下功能可能对解析返回的大量 CSV 数据很有用：

```
haProxyShowStats() {
   echo "show stat" | nc -U /var/run/haproxy.sock |sed '1s/^# *//'|
      awk -F',' -v fmt="%4s %12s %10s %6s %6s %4s %4s\n" '
         { if (NR==1) for (i=1;i<=NF;i++) f[tolower($i)]=i }
         { printf fmt, $f["sid"],$f["pxname"],$f["svname"],$f["status"],
                       $f["weight"],$f["act"],$f["bck"] }'
}
```

1. 最初，您的 haproxy 配置文件类似于以下内容：

```
server server1 host1:7054 check
server server2 host2:7054 check
```

将此配置更改为以下内容：

```
server server1 host1:7054 check backup
server server2 host2:7054 check backup
server server3 host3:7054 check
server server4 host4:7054 check
```

2. 使用新配置重新启动 HA 代理，如下所示：

```
haproxy -f <configfile> -st $(pgrep haproxy)
```

"haProxyShowStats" 现在将反映修改后的配置，其中包括两个活动的旧版本备份服务器和两个（尚未启动）升级的服务器：

```
sid   pxname      svname  status  weig  act  bck
  1   fabric-cas  server3   DOWN     1    1    0
  2   fabric-cas  server4   DOWN     1    1    0
  3   fabric-cas  server1     UP     1    0    1
  4   fabric-cas  server2     UP     1    0    1
```

3. 在 host3 和 host4 上安装 fabric-ca-server 的升级二进制文件。主机 3 和主机 4 上的新升级服务器应配置为使用与主机 1 和主机 2 上的旧服务器相同的数据库。启动升级的服务器后，数据库将自动迁移。

haproxy 将所有新流量转发到已升级的服务器，因为它们没有配置为备份服务器。 在继续操作之前，请使用以下命令验证您的集群仍在正常运行。此外，现在应反映出所有服务器都处于活动状态，类似于以下内容：`"fabric-ca-client getcainfo""haProxyShowStats"`

```
sid   pxname      svname  status  weig  act  bck
  1   fabric-cas  server3    UP     1    1    0
  2   fabric-cas  server4    UP     1    1    0
  3   fabric-cas  server1    UP     1    0    1
  4   fabric-cas  server2    UP     1    0    1
```

4. 停止 host1 和 host2 上的旧服务器。在继续操作之前，请使用命令验证 新群集是否仍在正常运行。然后从 haproxy 配置文件中删除较旧的服务器备份配置，使其看起来类似于以下内容：`"fabric-ca-client getcainfo"`

```
server server3 host3:7054 check
server server4 host4:7054 check
```

5. 使用新配置重新启动 HA 代理，如下所示：

```
haproxy -f <configfile> -st $(pgrep haproxy)
```

`"haProxyShowStats"` 现在将反映修改后的配置，其中两个活动服务器已升级到新版本：

```
sid   pxname      svname   status   weig   act   bck
  1   fabric-cas  server3    UP        1     1     0
  2   fabric-cas  server4    UP        1     1     0
```

9 营运服务

CA Server 托管一个 HTTP 服务器，该服务器提供 RESTful“操作” API。该 API 供操作员使用，而不是网络的管理员或“用户”使用。

该 API 公开了以下功能：

普罗米修斯（Prometheus）操作指标目标（配置时）

配置操作服务

操作服务需要两个基本配置：

要监听的**地址**和**端口**。用于身份验证和加密的 **TLS 证书**和**密钥**。请注意，**这些证书应由单独的专用 CA 生成**。不要使用为任何渠道的任何组织生成证书的 CA。

可以在 `operations` 服务器的配置文件部分中配置 CA 服务器：

```
operations:
  # host and port for the operations server
  listenAddress: 127.0.0.1:9443

  # TLS configuration for the operations endpoint
  tls:
    # TLS enabled
    enabled: true

    # path to PEM encoded server certificate for the operations server
    cert:
```

```
    file: tls/server.crt

  # path to PEM encoded server key for the operations server
  key:
    file: tls/server.key

  # require client certificate authentication to access all resources
  clientAuthRequired: false

  # paths to PEM encoded ca certificates to trust for client authentication
  clientRootCAs:
    files: []
```

该 listenAddress 键定义的主机和端口，运行服务器将侦听。如果服务器应侦听所有地址，则可以省略主机部分。本 tls 部分用于指示是否为操作服务启用了 TLS，服务的证书和私钥的位置以及客户端身份验证应信任的证书颁发机构根证书的位置。如果 clientAuthRequired 为 true，则将要求客户端提供用于认证的证书。

营运安全

由于操作服务专注于操作，并且有意与 Fabric 网络无关，因此它不使用成员资格服务提供者进行访问控制。取而代之的是，操作服务完全依赖具有客户端证书身份验证的双向 TLS。

强烈建议通过在生产环境中设置 clientAuthRequired to 的值来启用双向 TLS true。使用此配置，要求客户端提供有效的认证证书。如果客户端不提供证书，或者服务无法验证客户端的证书，则该请求将被拒绝。请注意，如果 clientAuthRequired 设置为 false，则客户端不需要提供证书。但是，如果他们这样做，并且服务无法验证证书，则该请求将被拒绝。

禁用 TLS 后，将绕过授权，并且可以连接到操作终结点的所有客户端都可以使用 API。

指标

Fabric CA 公开了可以洞察系统行为的指标。操作员和管理员可以使用此信息来更好地了解系统随着时间的推移如何运行。

配置指标

Fabric CA 提供了两种公开指标的方法：基于 Prometheus 的**拉**模型和基于 StatsD 的**推**模型。

普罗米修斯

典型的 Prometheus 部署通过从已检测目标公开的 HTTP 端点请求度量来刮取度量。由于 Prometheus 负责请求指标，因此被认为是拉动系统。

配置后，Fabric CA Server 将 `/metrics` 在操作服务上显示资源。要启用 Prometheus，请将服务器配置文件中的提供程序值设置为 `prometheus`。

```
metrics:
  provider: Prometheus
```

统计数据

StatsD 是一个简单的统计聚合守护程序。度量标准被发送到 `statsd` 守护程序，在此将其收集，汇总并推送到后端以进行可视化和警报。由于此模型需要仪器化的流程才能将指标数据发送到 StatsD，因此被认为是推送系统。

通过将指标提供程序设置为服务器配置文件 `statsd` 中的 `metrics` 部分中，可以将 CA Server 配置为将指标发送到 StatsD 。该 `statsd` 小节还必须配置 StatsD 守护程序的地址，要使用的网络类型（`tcp` 或 `udp`），以及发送度量标准的频率。`prefix` 可以指定一个可选的选项，以帮助区分度量标准的来源（例如，区分来自单独服务器的度量标准），该度量标准将优先于所有生成的度量标准。

```
metrics:
  provider: statsd
  statsd:
    network: udp
    address: 127.0.0.1:8125
    writeInterval: 10s
    prefix: server-0
```

要查看生成的不同指标，请查看 Metrics Reference。

4. Fabric CA 客户端

本节介绍如何使用 fabric-ca-client 命令。

Fabric CA 客户端的主目录确定如下：

- 如果设置了–home 命令行选项，请使用其值
- 否则，如果 `FABRIC_CA_CLIENT_HOME` 设置了环境变量，请使用其值
- 否则，如果 `FABRIC_CA_HOME` 设置了环境变量，请使用其值
- 否则，如果 `CA_CFG_PATH` 设置了环境变量，请使用其值
- 否则，使用 `$HOME/.fabric-ca-client`

以下说明假定客户端配置文件存在于客户端的主目录中。

1 注册引导程序标识

首先，如果需要，请在客户端配置文件中自定义 CSR（证书签名请求）部分。请注意，`csr.cn` 必须将字段设置为引导程序标识的 ID。默认 CSR 值如下所示：

```
csr:
  cn: <<enrollment ID>>
  key:
    algo: ecdsa
    size: 256
  names:
    - C: US
      ST: North Carolina
      L:
      O: Hyperledger Fabric
      OU: Fabric CA
  hosts:
   - <<hostname of the fabric-ca-client>>
  ca:
    pathlen:
    pathlenzero:
    expiry:
```

有关这些 CSR 字段的说明，请参见 CSR 字段。

然后运行命令以注册身份。例如，以下命令 通过调用在 7054 端口本地运行的 Fabric CA 服务器来注册 ID 为 **admin** 和密码为 **adminpw** 的身份。`fabric-ca-client enroll`

```
export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca/clients/admin
fabric-ca-client enroll -u http://admin:adminpw@localhost:7054
```

enroll 命令将注册证书 (ECert)，相应的私钥和 CA 证书链 PEM 文件存储在 Fabric CA 客户端 `msp` 目录的子目录中。您将看到指示 PEM 文件存储位置的消息。

2 注册新身份

执行注册请求的身份必须当前已注册，并且还必须具有适当的权限才能注册正在注册的身份的类型。

特别是，Fabric CA 服务器在注册过程中进行了三项授权检查，如下所示：

1. 注册服务商（即调用者）必须具有“ hf.Registrar.Roles”属性，该属性带有逗号分隔的值列表，其中一个值等于要注册的身份的类型；例如，如果注册服务商具有值为“ peer”的“ hf.Registrar.Roles”属性，则注册服务商可以注册对等类型的身份，但不能注册客户端，管理员或订购者的身份。
2. 注册服务商的隶属关系必须等于所注册身份的隶属关系或前缀。例如，关联为“ ab”的注册服务商可以注册具有关联“ abc”的身份，但可以不注册具有关联“ ac”的身份。如果身份要求具有根从属关系，则从属关系请求应为点（“。”），并且注册服务商还必须具有根从属关系。如果注册请求中未指定从属关系，则将为注册者提供从属关系。
3. 如果满足以下所有条件，则注册服务商可以使用属性注册身份：
   - 注册服务商可以注册具有前缀“ hf”的 Fabric CA 保留的属性。仅当注册服务商拥有该属性并且该属性是 hf.Registrar.Attributes 属性值的一部分时。此外，如果属性是列表类型，则正在注册的属性的值必须等于注册服务商拥有的值或该值的子集。如果属性的类型为布尔值，则仅当注册服务商的属性值为“ true”时，注册服务商才能注册该属性。
   - 注册自定义属性（即名称不以“ hf。开头”的任何属性）要求注册服务商具有“ hf.Registar.Attributes”属性，并且要注册的属性或模式的值。唯一受支持的模式是末尾带有“ *”的字符串。例如，“ ab *”是一种匹配所有以“ ab”开头的属性名称的模式。例如，如果注册服务商具有 hf.Registrar.Attributes = orgAdmin，则注册服务商可以添加或从身份中删除的唯一属性是“ orgAdmin”属性。
   - 如果请求的属性名称是“ hf.Registrar.Attributes”，则执行附加检查以查看此属性的请求值是否等于或“ hf.Registrar.Attributes”的注册商值的子集。为此，每个请求的值必须与“ hf.Registrar.Attributes”属性的注册商值中的值匹配。例如，如果“ hf.Registrar.Attributes”的注册商的值是“ ab *，xyz”，而请求的属性值是“ abc，xyz”，则这是有效的，因为“ abc”匹配“ ab *”和“ xyz”匹配注册商的“ xyz”值。

例子：

有效方案：

1. 如果注册商具有属性“ hf.Registrar.Attributes = ab *，xyz”并且正在注册属性“ abc”，则有效的“ abc”与“ ab *”匹配。
2. 如果注册服务商具有属性“ hf.Registrar.Attributes = ab *，xyz”并且正在注册属性“ xyz”，则这是有效的，因为“ xyz”与注册服务商的“ xyz”值匹配。
3. 如果注册服务商具有属性“ hf.Registrar.Attributes = ab *，xyz”，并且请求的属性值为“ abc，xyz”，则这是有效的，因为“ abc”与“ ab *”匹配，而“ xyz”与注册商的“ xyz的值。
4. 如果注册商的属性为“ hf.Registrar.Roles =对等，客户端，管理员，订购者”，并且请求的属性值为“对等”，“对等，客户端，管理员，订购者”或“客户端，管理员”，则该属性有效因为请求的值等于注册服务商的值或其一部分。

无效的方案：

1. 如果注册商具有属性“ hf.Registrar.Attributes = ab *，xyz”并且正在注册属性“ hf.Registar.Attributes = abc，xy *”，则该无效，因为请求的属性“ xy *”不是拥有的模式由注册商。值“ xy *”是“ xyz”的超集。

2. 如果注册服务商具有属性'hf.Registrar.Attributes = ab *，xyz'并且正在注册属性'hf.Registar.Attributes = abc，xyz，attr1'，则此无效，因为注册服务商的'hf.Registrar.Attributes'属性值不包含“ attr1”。
3. 如果注册服务商具有属性“ hf.Registrar.Attributes = ab *，xyz”并且正在注册属性“ a.b”，则该无效，因为值“ a.b”未包含在“ ab *”中。
4. 如果注册商具有属性“ hf.Registrar.Attributes = ab *，xyz”并且正在注册属性“ x.y”，则该无效，因为“ xyz”不包含“ x.y”。
5. 如果注册服务商的属性为“ hf.Registrar.Roles = peer”，并且请求的属性值为“ peer，client”，则此无效，因为注册服务商的 hf.Registrar.Roles 属性值不具有客户端角色。
6. 如果注册服务商具有属性“ hf.Revoker = false”，并且请求的属性值为“ true”，则该属性无效，因为 hf.Revoker 属性是布尔属性，并且注册商的属性值不是“ true”。

下表列出了可以为身份注册的所有属性。属性名称区分大小写。

注意：注册身份时，您可以指定属性名称和值的数组。如果数组指定了多个具有相同名称的数组元素，则当前仅使用最后一个元素。换句话说，当前不支持多值属性。

以下命令使用**管理员**身份的凭据注册一个新的身份，其注册 ID 为“ admin2”，从属关系为“ org1.department1”，一个名为“ hf.Revoker”的属性，其值为“ true”，以及一个属性名为“ admin”，值为“ true”。“：ecert”后缀表示默认情况下，“ admin”属性及其值将插入到身份的注册证书中，然后可用于制定访问控制决策。

```
export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca/clients/admin
fabric-ca-client register --id.name admin2 --id.affiliation org1.department1 --id.attrs
'hf.Revoker=true,admin=true:ecert'
```

密码（也称为注册密码）被打印出来。需要此密码才能注册身份。这允许管理员注册一个身份，并将注册 ID 和秘密提供给其他人以注册该身份。
可以将多个属性指定为–id.attrs 标志的一部分，每个属性必须用逗号分隔。对于包含逗号的属性值，必须将该属性封装在双引号中。请参见下面的示例。

```
fabric-ca-client register -d --id.name admin2 --id.affiliation org1.department1 --id.attrs
'"hf.Registrar.Roles=peer,client",hf.Revoker=true'
```

要么

```
fabric-ca-client register -d --id.name admin2 --id.affiliation org1.department1 --id.attrs
'"hf.Registrar.Roles=peer,client"' --id.attrs hf.Revoker=true
```

您可以通过编辑客户端的配置文件为 register 命令中使用的任何字段设置默认值。例如，假设配置文件包含以下内容：

```
id:
  name:
  type: client
  affiliation: org1.department1
  maxenrollments: -1
  attributes:
    - name: hf.Revoker
      value: true
    - name: anotherAttrName
```

```
      value: anotherAttrValue
```

然后，以下命令将从命令行获取一个新的标识，其注册 ID 为“ admin3”，其余的则从配置文件中获取，包括标识类型：“ client”，隶属关系：“ org1.department1” ，以及两个属性：“ hf.Revoker”和“ anotherAttrName”。

```
export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca/clients/admin
fabric-ca-client register --id.name admin3
```

要注册具有多个属性的身份，需要如上所述在配置文件中指定所有属性名称和值。
将 *maxenrollments* 设置为 0 或将其保留在配置之外，将导致身份被注册为使用 CA 的最大注册值。此外，正在注册的身份的最大注册值不能超过 CA 的最大注册值。例如，如果 CA 的最大注册值为 5，则任何新标识的值都必须小于或等于 5，并且也不能将其设置为-1（无限注册）。
接下来，让我们注册一个对等方身份，该身份将在以下部分中用于注册对等方。以下命令注册 **peer1** 身份。请注意，我们选择指定自己的密码（或密码），而不是让服务器为我们生成一个密码。

```
export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca/clients/admin
fabric-ca-client register --id.name peer1 --id.type peer --id.affiliation org1.department1 --
id.secret peer1pw
```

请注意，从属关系区分大小写，但服务器配置文件中指定的非叶子从属关系始终以小写形式存储。例如，如果服务器配置文件的从属关系部分如下所示：

```
affiliations:
  BU1:
    Department1:
      - Team1
  BU2:
    - Department2
    - Department3
```

BU1，*Department1*，*BU2* 以小写形式存储。这是因为 Fabric CA 使用 Viper 读取配置。Viper 将映射键视为不区分大小写，并且始终返回小写值。要注册具有标识 *TEAM1* 从属关系，*bu1.department1.Team1* 将需要被指定给 *-id.affiliation* 标志如下所示：

```
export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca/clients/admin
fabric-ca-client register --id.name client1 --id.type client --id.affiliation bu1.department1.Team1
```

3 注册对等身份

现在您已经成功注册了对等方身份，您现在可以根据给定的注册 ID 和密码（即 上一节中的*密码）*来注册对等方。这与注册引导程序标识相似，除了我们还将演示如何使用“ -M”选项填充 Hyperledger Fabric MSP（成员资格服务提供程序）目录结构。

以下命令注册了 peer1。确保将“ -M”选项的值替换为对等方的 MSP 目录的路径，该路径是对等方的 core.yaml 文件中的“ mspConfigPath”设置。您也可以将 FABRIC_CA_CLIENT_HOME 设置为对等方的主目录。

```
export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca/clients/peer1
```

```
fabric-ca-client enroll -u http://peer1:peer1pw@localhost:7054 -M $FABRIC_CA_CLIENT_HOME/msp
```

注册排序者是相同的，除了 MSP 目录的路径是订购者的 orderer.yaml 文件中的“ LocalMSPDir”设置。
由 fabric-ca-server 颁发的所有注册证书均具有以下组织单位（或简称为 “ OU” ）：

1. OU 层次结构的根等于身份类型
2. 为身份隶属关系的每个组成部分添加一个 OU

例如，如果一个标识的类型为*对等体*和其从属关系是 *department1.team1*，身份的 OU 层次结构（从叶根）是 *OU = TEAM1*，*OU = department1*，*OU =对等体*。

4 获取身份混合器凭证

身份混合器（Idemix）是一种加密协议套件，用于保护隐私的身份验证和认证属性的传输。Idemix 允许客户在没有发行人（CA）参与的情况下向验证者进行身份验证，并选择性地仅公开验证者所需的那些属性，并且可以这样做，而无需跨其交易进行链接。

除 X509 证书外，Fabric CA 服务器还可以颁发 Idemix 凭据。可以通过将请求发送到 `/api/v1/idemix/credential` API 端点来请求 Idemix 凭据。有关此和其他 Fabric CA 服务器 API 端点的更多信息，请参考 swagger-fabric-ca.json。

Idemix 证书颁发过程分为两个步骤。首先，将带有空主体的请求发送到 `/api/v1/idemix/credential` API 端点，以获取现时和 CA 的 Idemix 公钥。其次，使用随机数和 CA 的 Idemix 公钥创建凭据请求，然后将主体中带有凭据请求的另一个请求发送到 `/api/v1/idemix/credential` API 端点，以获取 Idemix 凭据，凭据吊销信息（CRI）以及属性名称和值。当前，仅支持三个属性：

- **OU**-身份的组织单位。此属性的值设置为标识的从属关系。例如，如果标识的从属关系是 *dept1.unit1*，则 OU 属性设置为 *dept1.unit1*
- **IsAdmin**-身份是否为管理员。该属性的值设置为 *isAdmin* 注册属性的值。
- **EnrollmentID**-身份的注册 ID

您可以参考 https://github.com/hyperledger/fabric-ca/blob/master/lib/client.go 中的 *handleIdemixEnroll* 函数，以获得获取 Idemix 凭据的两步过程的参考实现。

该 `/api/v1/idemix/credential` API 端点接受的基本和令牌授权头。基本授权标头应包含用户的注册 ID 和密码。如果身份已经具有 X509 注册证书，则它也可以用于创建令牌授权标头。

请注意，Hyperledger Fabric 将支持客户端使用 X509 和 Idemix 凭据对交易进行签名，但仅支持 X509 凭据用于对等身份和订购者身份。和以前一样，应用程序可以使用 Fabric SDK 将请求发送到 Fabric CA 服务器。SDK 隐藏了与创建授权标头和请求有效负载以及处理响应相关的复杂性。

5. 获取 Idemix CRI（证书吊销信息）

Idemix CRI（证书吊销信息）的目的类似于 X509 CRL（证书吊销列表）：吊销以前发布的证书。但是，有一些差异。在 X509 中，发行者吊销最终用户的证书，并且其 ID 包含在 CRL 中。验证程序检查用户的证书是否在 CRL 中，如果是，则返回授权失败。最终用户除了从验证程序收到授权错误外，不参与此吊销过程。
在 Idemix 中，涉及最终用户。发行者撤销了类似于 X509 的最终用户的凭证，并且该撤销的证据记录在 CRI 中。将 CRI 提供给最终用户（也称为“提供者”）。然后，最终用户会生成证明其凭证尚未根据 CRI 撤销。最终用户将此证明提供给验证者，验证者根据 CRI 验证该证明。为了使验证成功，最终用户和验证者使用的 CRI 版本（称为“纪元”）必须相同。可以通过向 `/api/v1/idemix/cri` API 端点发送请求来请求最新的 CRI 。
当 fabric-ca-server 收到注册请求并且吊销句柄池中没有吊销句柄时，CRI 的版本将增加。在这种情况下，fabric-ca-server 必须生成一个新的吊销句柄池，以增加 CRI 的时间。可通过 `idemix.rhpoolsize` 服务器配置属性来配置撤销句柄池中的撤销句柄数。

1 重新注册身份

假设您的注册证书即将过期或已被盗用。您可以发出 reenroll 命令来更新您的注册证书，如下所示。

```
export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca/clients/peer1
fabric-ca-client reenroll
```

2 吊销证书或身份

身份或证书可以被撤销。撤消身份将撤消该身份拥有的所有证书，并且还将阻止该身份获取任何新证书。吊销证书将使单个证书无效。
为了撤消证书或身份，主叫身份必须具有 `hf.Revoker` and `hf.Registrar.Roles` 属性。吊销身份只能吊销证书或身份与该吊销身份的隶属关系相同或以其为前缀的身份。此外，revoker 只能撤消具有 revoker `hf.Registrar.Roles` 属性中列出的类型的身份 。
例如，具有 **orgs.org1** 和'hf.Registrar.Roles = peer，client'属性的吊销者可以吊销与 **orgs.org1** 或 **orgs.org1.department1** 关联的**对等**或**客户端**类型身份， 但不能吊销身份与 **orgs.org2** 相关联 或任何其他类型。

以下命令禁用身份并吊销与该身份关联的所有证书。Fabric CA 服务器从该身份接收到的所有将来的请求都将被拒绝。

```
fabric-ca-client revoke -e <enrollment_id> -r <reason>
```

以下是可以使用 `-r` flag 指定的受支持原因：

1. 未指定
2. 关键妥协
3. 妥协
4. 隶属关系变更

5. 被取代
6. 停止操作
7. 证书持有人
8. removefromcrl
9. 特权撤销
10. 妥协

例如，与从属树的根相关联的引导管理员可以撤消 **peer1** 的身份，如下所示：

```
export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca/clients/admin
fabric-ca-client revoke -e peer1
```

可以通过指定其 AKI（授权密钥标识符）和序列号来撤销属于身份的注册证书，如下所示：

```
fabric-ca-client revoke -a xxx -s yyy -r <reason>
```

例如，您可以使用 openssl 命令获取证书的 AKI 和序列号，并将它们传递给 `revoke` 命令以撤消所述证书，如下所示：

```
serial=$(openssl x509 -in userecert.pem -serial -noout | cut -d "=" -f 2)
aki=$(openssl x509 -in userecert.pem -text | awk '/keyid/ {gsub(/ *keyid:|:/,"",$1);print
tolower($0)}')
fabric-ca-client revoke -s $serial -a $aki -r affiliationchange
```

该 *-gencrl* 标志可以被用来生成一个包含所有已撤消证书一 CRL（证书吊销列表）。例如，以下命令将吊销标识 **peer1**，生成一个 CRL 并将其存储在**<msp folder> /crls/crl.pem** 文件中。

```
fabric-ca-client revoke -e peer1 --gencrl
```

也可以使用 *gencrl* 命令生成 CRL 。 有关 *gencrl* 命令的更多信息，请参考生成 CRL（证书吊销列表）部分。

3 生成 CRL（证书吊销列表）

在 Fabric CA 服务器中吊销证书后，还必须更新 Hyperledger Fabric 中的相应 MSP。这包括对等方的本地 MSP 以及相应通道配置块中的 MSP。为此，必须将 PEM 编码的 CRL（证书吊销列表）文件放在 MSP 的 *crls* 文件夹中。该命令可用于生成 CRL。具有属性的任何标识都可以创建一个 CRL，其中包含在一定时期内被吊销的所有证书的序列号。创建的 CRL 存储在 *<msp 文件夹> /crls/crl.pem* 文件中。`fabric-ca-client gencrlhf.GenCRL`

以下命令将创建一个 CRL，其中包含所有已撤销的证书（过期和未过期），并将 CRL 存储在 *~/msp / crls / crl.pem* 文件中。

```
export FABRIC_CA_CLIENT_HOME=~/clientconfig
fabric-ca-client gencrl -M ~/msp
```

下一条命令将创建一个 CRL，其中包含 2017-09-13T16：39：57-08：00（由*–revokedafter* 标志指定）和 2017-09-21T16：39 之前吊销的所有证书（已过期和未过期）： 57-08：00（由*–revokedbefore* 标志指定），并将 CRL 存储在 *~/ msp / crls / crl.pem* 文件中。

```
export FABRIC_CA_CLIENT_HOME=~/clientconfig
fabric-ca-client gencrl --caname "" --revokedafter 2017-09-13T16:39:57-08:00 --revokedbefore 2017-
09-21T16:39:57-08:00 -M ~/msp
```

该 *-caname* 标志指定到该请求被发送的 CA 的名称。在此示例中，gencrl 请求被发送到默认 CA。

的*-revokedafter*和*-revokedbefore*标志指定的时间段的下边界和上边界。生成的 CRL 将包含在此时间段内被吊销的证书。该值必须是以 RFC3339 格式指定的 UTC 时间戳。该*-revokedafter*时间戳不能比越大 -*revokedbefore*时间戳。

默认情况下，CRL 的“下一次更新”日期设置为第二天。该 *crl.expiry* CA 配置属性可以用来指定自定义值。

gencrl 命令还将接受*–expireafter*和*–expirebefore*标志，*这些*标志可用于生成带有已撤销证书的 CRL，这些证书在这些标志指定的时间段内到期。例如，以下命令将生成一个 CRL，其中包含在 2017-09-13T16：39：57-08：00 之后和 2017-09-21T16：39：57-08：00 之前被吊销的证书，这些证书在 2017-09-13T16：39：57-08：00 和 2018-09-13T16：39：57-08：00 之前

```
export FABRIC_CA_CLIENT_HOME=~/clientconfig
fabric-ca-client gencrl --caname "" --expireafter 2017-09-13T16:39:57-08:00 --expirebefore 2018-09-13T16:39:57-08:00  --revokedafter 2017-09-13T16:39:57-08:00 --revokedbefore 2017-09-21T16:39:57-08:00 -M ~/msp
```

4 启用 TLS

本节更详细地描述如何为 Fabric CA 客户端配置 TLS。

可以在中配置以下部分 fabric-ca-client-config.yaml 。

```
tls:
  # Enable TLS (default: false)
  enabled: true
  certfiles:
    - root.pem
  client:
    certfile: tls_client-cert.pem
    keyfile: tls_client-key.pem
```

该 **certfiles** 选项是一组由客户端受信任的根证书。通常，这只是在 **ca-cert.pem** 文件的服务器主目录中找到的根 Fabric CA 服务器的证书。

仅当在服务器上配置了双向 TLS 时，才需要使用 **client** 选项。

5 基于属性的访问控制

访问控制决策可以由链码 (以及 Hyperledger Fabric 运行时) 基于身份的属性来做出。这称为 **基于属性的访问控制**，或简称 **ABAC**。

为了使之成为可能，身份的注册证书（ECert）可能包含一个或多个属性名称和值。然后，链码提取属性的值以做出访问控制决策。

例如，假设您正在开发应用程序 *app1*，并且希望只有 app1 管理员才能访问特定的链码操作。你可以 chaincode 核实来电者的证书（颁发由 CA 信任的通道）包含一个名为属性 *app1Admin*，值为*真*。当然，属性的名称可以是任何值，并且该值不必是布尔值。

那么，如何获得具有属性的注册证书？有两种方法：

1. 注册身份时，可以指定为该身份颁发的注册证书默认情况下应包含一个属性。可以在注册时覆盖此行为，但这对于建立默认行为很有用，并且假设注册发生在您的应用程序外部，则不需要任何应用程序更改。

   下面显示了如何使用两个属性注册 *user1*： *app1Admin* 和 *email*。当用户未在注册时显式请求属性时，后缀“：ecert”会使 *appAdmin* 属性默认插入到 user1 的注册证书中。默认情况下，*电子邮件*属性不会添加到注册证书中。

```
fabric-ca-client register --id.name user1 --id.secret user1pw --id.type client --id.affiliation
org1 --id.attrs 'app1Admin=true:ecert,email=user1@gmail.com'
```

2. 注册身份时，您可以明确要求将一个或多个属性添加到证书中。对于每个请求的属性，您可以指定该属性是否为可选。如果不是可选请求，并且身份不具有该属性，则将发生错误。

   下面显示了如何使用 *email* 属性，不带 *app1Admin* 属性，以及可选*地带 phone* 属性（如果用户拥有 *phone* 属性）注册 *user1*。

```
fabric-ca-client enroll -u http://user1:user1pw@localhost:7054 --enrollment.attrs "email,phone:opt"
```

下表显示了针对每个身份自动注册的三个属性。

属性名称	属性值
hf.EnrollmentID	身份的注册 ID
hf.Type	身份类型
会员关系	身份的从属关系

要将上述任何属性**默认添加**到证书中，必须使用“：ecert”规范明确注册该属性。例如，以下代码注册身份“ user1”，以便在注册时不要求特定属性的情况下，将“ hf.Affiliation”属性添加到注册证书中。请注意，“– id.affiliation”和“ –id.attrs”标志中的从属关系值（即“ org1”）必须相同。

```
fabric-ca-client register --id.name user1 --id.secret user1pw --id.type client --id.affiliation org1
--id.attrs 'hf.Affiliation=org1:ecert'
```

有关基于属性的访问控制的链码库 API 的信息，请参见 https://github.com/hyperledger/fabric-chaincode-go/blob/master/pkg/cid/README.md

6 动态服务器配置更新

本节介绍如何使用 fabric-ca-client 动态更新 Fabric-ca-server 配置的某些部分而无需重新启动服务器。

本节中的所有命令都要求您首先通过执行 *fabric-ca-client enroll* 命令进行*注册*。

动态更新身份

本节介绍如何使用 fabric-ca-client 动态更新身份。

如果客户端身份不满足以下所有条件，则会发生授权失败：

- 客户端身份必须具有“ hf.Registrar.Roles”属性，该属性带有用逗号分隔的值列表，其中一个值等于要更新的身份的类型；例如，如果客户端的身份具有值为“ client”的“ hf.Registrar.Roles”属性，则客户端可以更新“客户端”类型的身份，但不能更新“对等”类型。
- 客户身份的从属关系必须等于要更新的身份的从属关系或前缀。例如，具有“ ab”的从属的客户端可以以“ abc”的从属关系更新身份，但是可以不具有“ ac”的从属关系更新身份。如果身份要求具有根关系，则更新请求应为该关系指定一个点（“。”），并且客户端也必须具有根关系。

下面显示了如何添加，修改和删除从属关系。

获取身份信息

只要呼叫者满足以上部分中突出显示的授权要求，呼叫者就可以从 Fabric-ca 服务器检索有关身份的信息。以下命令显示如何获取身份。

```
fabric-ca-client identity list --id user1
```

调用方还可以通过发出以下命令来请求检索有关已授权查看的所有身份的信息。

```
fabric-ca-client identity list
```

添加身份

以下内容为“ user1”添加了新的标识。添加新身份与通过“ fabric-ca-client register”命令注册身份的操作相同。有两种可用的添加新身份的方法。第一种方法是通过 *–json* 标志，您可以在其中描述 JSON 字符串中的身份。

```
fabric-ca-client identity add user1 --json '{"secret": "user1pw", "type": "client", "affiliation":
"org1", "max_enrollments": 1, "attrs": [{"name": "hf.Revoker", "value": "true"}]}'
```

以下内容添加了具有根从属关系的用户。请注意，从属名称“。”表示根从属。

```
fabric-ca-client identity add user1 --json '{"secret": "user1pw", "type": "client", "affiliation":
".", "max_enrollments": 1, "attrs": [{"name": "hf.Revoker", "value": "true"}]}'
```

添加身份的第二种方法是使用直接标志。请参见下面的示例以添加“ user1”。

```
fabric-ca-client identity add user1 --secret user1pw --type client --affiliation . --maxenrollments
1 --attrs hf.Revoker=true
```

下表列出了身份的所有字段，它们是必填字段还是可选字段，以及它们可能具有的任何默认值。

Fields	Required	Default Value
ID	Yes	
Secret	No	
Affiliation	No	Caller’s Affiliation
Type	No	client
Maxenrollments	No	0
Attributes	No	

修改身份

有两种可用的修改现有身份的方法。第一种方法是通过 *–json* 标志，您可以在其中描述对 JSON 字符串中的身份*所做*的修改。可以在单个请求中进行多次修改。身份的任何未经修改的元素将保留其原始值。

注意：maxenrollments 值为 “ -2” 指定将使用 CA 的最大注册设置。

下面的命令使用–json 标志对身份进行多次修改。

```
fabric-ca-client identity modify user1 --json '{"secret": "newPassword", "affiliation": ".",
"attrs": [{"name": "hf.Regisrar.Roles", "value": "peer,client"},{"name": "hf.Revoker", "value":
"true"}]}'
```

下面的命令使用直接标志进行修改。以下内容将身份 “ user1” 的注册机密（或密码）更新为 “ newsecret” 。

```
fabric-ca-client identity modify user1 --secret newsecret
```

以下内容将标识“ user1”的从属关系更新为“ org2”。

```
fabric-ca-client identity modify user1 --affiliation org2
```

身份 修改 用户 1 - 隶属关系 ORG2

以下将身份'user1'的类型更新为'peer'。

```
fabric-ca-client identity modify user1 --type peer
```

以下内容将身份 “ user1” 的最大注册人数更新为 5。

```
fabric-ca-client identity modify user1 --maxenrollments 5
```

通过将 maxenrollments 值指定为 “ -2” ，将导致身份 “ user1” 使用 CA 的最大注册设置。

```
fabric-ca-client identity modify user1 --maxenrollments -2
```

以下将身份 “ user1” 的 “ hf.Revoker” 属性的值设置为 “ false” 。如果身份具有其他属性，则不会更改它们。如果该身份先前不具有 “ hf.Revoker” 属性，则将该属性添加到该身份。也可以通过不为属性指定任何值来删除属性。

```
fabric-ca-client identity modify user1 --attrs hf.Revoker=false
```
以下内容删除了用户'user1'的'hf.Revoker'属性。

```
fabric-ca-client identity modify user1 --attrs hf.Revoker=
```
以下内容演示了在单个 *fabric-ca-client 身份修改* 命令中可以使用多个选项。在这种情况下，将为用户“ user1”更新密码和类型。

```
fabric-ca-client identity modify user1 --secret newpass --type peer
```

删除身份

以下内容删除身份“ user1”，并吊销与“ user1”身份相关的所有证书。

```
fabric-ca-client identity remove user1
```
注意：默认情况下，在 fabric-ca-server 中禁用身份删除，但是可以通过使用*–cfg.identities.allowremove* 选项启动 Fabric-ca-server 来启用。

动态更新从属关系

本节介绍如何使用 fabric-ca-client 动态更新从属关系。下面显示了如何添加，修改，删除和列出从属关系。

添加从属关系

如果客户端身份不满足以下所有条件，则会发生授权失败：

- 客户身份必须具有值为“ true”的属性“ hf.AffiliationMgr”。
- 客户身份的从属关系必须在层次上高于要更新的从属关系。例如，如果客户的从属关系是“ ab”，则客户可以添加从属关系“ abc”，而不是“ a”或“ ab”。

以下内容添加了一个名为“ org1.dept1”的新隶属关系。

```
fabric-ca-client affiliation add org1.dept1
```
修改从属关系

如果客户端身份不满足以下所有条件，则会发生授权失败：

- 客户身份必须具有值为“ true”的属性“ hf.AffiliationMgr”。
- 客户身份的从属关系必须在层次上高于要更新的从属关系。例如，如果客户的从属关系是“ ab”，则客户可以添加从属关系“ abc”，而不是“ a”或“ ab”。
- 如果“ –force”选项为 true，并且必须修改身份，则还必须授权客户端身份来修改身份。

以下将“ org2”从属关系重命名为“ org3”。它还重命名了所有子从属关系（例如，“ org2.department1”重命名为“ org3.department1”）。

```
fabric-ca-client affiliation modify org2 --name org3
```

如果某些身份受关联的重命名影响，除非使用“ -force”选项，否则将导致错误。使用“ –force”选项将更新受影响的身份的从属关系，以使用新的从属名称。

```
fabric-ca-client affiliation modify org1 --name org2 –force
```

删除从属关系

如果客户端身份不满足以下所有条件，则会发生授权失败：

- 客户身份必须具有值为“ true”的属性“ hf.AffiliationMgr”。
- 客户身份的从属关系必须在层次上高于要更新的从属关系。例如，如果客户的从属关系是“ ab”，则客户可以删除从属关系“ abc”，而不是“ a”或“ ab”。
- 如果“ –force”选项为 true，并且必须修改身份，则还必须授权客户端身份来修改身份。

以下内容删除从属关系“ org2”以及所有子从属关系。例如，如果“ org2.dept1”是“ org2”以下的从属关系，则也会将其删除。

```
fabric-ca-client affiliation remove org2
```

如果某些身份受关联关系的删除影响，除非使用“ -force”选项，否则将导致错误。使用“ –force”选项还将删除与该从属关系关联的所有身份，以及与这些身份中的任何一个关联的证书。

注意：默认情况下，Fabric-ca-server 中的联盟关系删除是禁用的，但可以通过使用–*cfg.affiliations.allowremove* 选项启动 Fabric-ca-server 来启用。

列出从属会员信息

如果客户端身份不满足以下所有条件，则会发生授权失败：

- 客户身份必须具有值为“ true”的属性“ hf.AffiliationMgr”。
- 客户身份的从属关系必须等于或在层次上高于要更新的从属关系。例如，如果客户的隶属关系是“ ab”，则客户可能会获得关于“ ab”或“ abc”的隶属信息，而不是“ a”或“ ac”。

以下命令显示如何获取特定的从属关系。

```
fabric-ca-client affiliation list --affiliation org2.dept1
```

调用方还可以通过发出以下命令来请求检索有关其有权查看的所有从属关系的信息。

```
fabric-ca-client affiliation list
```

7 管理证书

本节介绍如何使用 fabric-ca-client 管理证书。

列出证书信息

呼叫者可见的证书包括：

- 那些属于调用者的证书
- 如果呼叫者拥有 `hf.Registrar.Roles` 属性或 `hf.Revoker` 值为的属性 `true`，则所有属于呼叫者所属关系之内或之下的身份的证书。例如，如果客户端的从属关系 `a.b`，客户端可能会得到身份证明谁的从属关系 `a.b` 或 `a.b.c` 而不是 `a` 或 `a.c`。

如果执行请求多于一个身份的证书的列表命令，则仅列出其亲属关系等于或在层次上低于呼叫者的亲属关系的身份证明书。

可以根据 ID，AKI，序列号，到期时间，吊销时间，未吊销和未过期标志过滤将列出的证书。

- `id`：列出此注册 ID 的证书
- `serial`：列出具有此序列号的证书
- `aki`：列出具有此 AKI 的证书
- `expiration`：列出过期日期在此过期时间内的证书
- `revocation`：列出在此撤销时间内被撤销的证书
- `notrevoked`：列出尚未撤销的证书
- `notexpired`：列出尚未过期的证书

您可以使用标志 `notexpired` 并将其 `notrevoked` 用作过滤器，以从结果集中排除吊销的证书和/或过期的证书。例如，如果您只关心已过期但尚未撤消的证书，则可以使用 `expiration` 和 `notrevoked` 标志获取此类结果。下面提供了这种情况的示例。

时间应基于 RFC3339 指定。例如，要列出在 2018 年 3 月 1 日下午 1:00 到 2018 年 6 月 15 日下午 2:00 之间过期的证书，输入时间字符串应类似于 2018-03-01T13：00：00z 和 2018-06 -15T02：00：00z。如果不关心时间，仅考虑日期，则可以省略时间部分，然后字符串变为 2018-03-01 和 2018-06-15。

该字符串 `now` 可用于表示当前时间，空字符串可用于表示任何时间。例如，`now::` 表示从现在到将来的任何时间 `::now` 的时间范围，并且表示从过去到现在的任何时间的时间范围。

以下命令显示如何使用各种过滤器列出证书。

列出所有证书：

```
fabric-ca-client certificate list
```

按 ID 列出所有证书：

```
fabric-ca-client certificate list --id admin
```

按序列号和 aki 列出证书：

```
fabric-ca-client certificate list --serial 1234 --aki 1234
```

按 ID 和序列号/ aki 列出证书：

```
fabric-ca-client certificate list --id admin --serial 1234 --aki 1234
```

列出既不是撤销者也不是 ID 过期的证书：

```
fabric-ca-client certificate list --id admin --notrevoked --notexpired
```

列出所有尚未被撤销 ID（管理员）的证书：

```
fabric-ca-client certificate list --id admin --notrevoked
```

列出所有未过期的 ID 证书（管理员）：

“–notexpired”标志等同于“ --expiration now ::”，这意味着证书将在将来的某个时间到期。

```
fabric-ca-client certificate list --id admin --notexpired
```

列出在一个 ID 范围内（管理员）被吊销的所有证书：

```
fabric-ca-client certificate list --id admin --revocation 2018-01-01T01:30:00z::2018-01-30T05:00:00z
```

列出在某个时间范围内被吊销但尚未过期的所有证书（管理员）：

```
fabric-ca-client certificate list --id admin --revocation 2018-01-01::2018-01-30 --notexpired
```

使用持续时间（在 30 天到 15 天之间撤销）列出 ID（管理员）列出所有已撤销的证书：

```
fabric-ca-client certificate list --id admin --revocation -30d::-15d
```

列出所有撤销的证书

```
fabric-ca-client certificate list --revocation ::2018-01-30
```

列出一段时间后所有已撤销的证书

```
fabric-ca-client certificate list --revocation 2018-01-30::
```

列出现在之前和特定日期之后的所有吊销证书

```
fabric-ca-client certificate list --id admin --revocation 2018-01-30::now
```

列出所有在某个时间范围内到期但尚未撤销 ID（管理员）的证书：

```
fabric-ca-client certificate list --id admin --expiration 2018-01-01::2018-01-30 --notrevoked
```

使用持续时间（在 30 天到 15 天之间的过期时间）列出 ID（管理员）列出所有已过期的证书：

```
fabric-ca-client certificate list --expiration -30d::-15d
```

列出所有已过期或将在特定时间之前过期的证书

```
fabric-ca-client certificate list --expiration ::2058-01-30
```

列出所有已过期或将在一定时间后过期的证书

```
fabric-ca-client certificate list --expiration 2018-01-30::
```
列出现在之前和特定日期之后的所有过期证书

```
fabric-ca-client certificate list --expiration 2018-01-30::now
```
列出在接下来的 10 天内到期的证书：

```
fabric-ca-client certificate list --id admin --expiration ::+10d --notrevoked
```
list certificate 命令还可以用于在文件系统上存储证书。这是在 MSP 中填充 admins 文件夹的便捷方法。“-store”标志指向文件系统上用于存储证书的位置。

通过在 MSP 中存储用于身份的证书，将身份配置为管理员：

```
export FABRIC_CA_CLIENT_HOME=/tmp/clientHome
fabric-ca-client certificate list --id admin --store msp/admincerts
```

8 联系特定的 CA 实例

当服务器运行多个 CA 实例时，可以将请求定向到特定的 CA。默认情况下，如果客户端请求中未指定 CA 名称，则该请求将定向到 fabric-ca 服务器上的默认 CA。可以使用 `caname` 过滤器在客户端命令的命令行上指定 CA 名称，如下所示：

```
fabric-ca-client enroll -u http://admin:adminpw@localhost:7054 --caname <caname>
```

6. 配置 HSM

默认情况下，Fabric CA 服务器和客户端将私钥存储在 PEM 编码的文件中，但也可以将它们配置为通过 PKCS11 API 将私钥存储在 HSM（硬件安全模块）中。在服务器或客户端的配置文件的 BCCSP（BlockChain 加密服务提供程序）部分中配置了此行为。当前，Fabric 仅支持与 HSM 通信的 PKCS11 标准。

1 例

以下示例演示如何配置 Fabric CA 服务器或客户端以使用称为 SoftHSM 的 PKCS11 软件版本（请参阅 https://github.com/opendnssec/SoftHSMv2）。

安装 SoftHSM 之后，请确保将 SOFTHSM2_CONF 环境变量设置为指向 SoftHSM2 配置文件的存储位置。配置文件看起来像

```
directories.tokendir = /tmp/
objectstore.backend = file
log.level = INFO
```

您可以在 testdata 目录下找到名为 SoftHSM2.conf 的示例配置文件。

创建一个令牌，将其标记为“ ForFabric”，将引脚设置为“ 98765432”（请参阅 SoftHSM 文档）。

您可以同时使用配置文件和环境变量来配置 BCCSP。例如，按如下所示设置 Fabric CA 服务器配置文件的 bccsp 部分。请注意，默认字段的值为 PKCS11。

```
###############################################################################
# BCCSP (BlockChain Crypto Service Provider) section is used to select which
# crypto library implementation to use
###############################################################################
bccsp:
  default: PKCS11
  pkcs11:
    Library: /usr/local/Cellar/softhsm/2.1.0/lib/softhsm/libsofthsm2.so
    Pin: 98765432
    Label: ForFabric
    hash: SHA2
    security: 256
    filekeystore:
      # The directory used for the software file-based keystore
      keystore: msp/keystore
```

您可以通过环境变量覆盖相关字段，如下所示：

```
FABRIC_CA_SERVER_BCCSP_DEFAULT=PKCS11
FABRIC_CA_SERVER_BCCSP_PKCS11_LIBRARY=/usr/local/Cellar/softhsm/2.1.0/lib/softhsm/libsofthsm2.so
FABRIC_CA_SERVER_BCCSP_PKCS11_PIN=98765432
FABRIC_CA_SERVER_BCCSP_PKCS11_LABEL=ForFabric
```

预先构建的 Hyperledger Fabric Docker 映像未启用以使用 PKCS11。如果要使用 Docker 部署 Fabric CA，则需要构建自己的映像并使用以下命令启用 PKCS11：

```
make docker GO_TAGS=pkcs11
```

您还需要通过安装 PKCS11 库或将其安装在容器中来确保 CA 可以使用 PKCS11 库。如果要使用 Docker Compose 部署 Fabric CA，则可以更新 Compose 文件以使用卷将 SoftHSM 库和配置文件安装在容器内。例如，您将以下环境和卷变量添加到 Docker Compose 文件中：

```
environment:
   - SOFTHSM2_CONF=/etc/hyperledger/fabric/config.file
volumes:
   - /home/softhsm/config.file:/etc/hyperledger/fabric/config.file
   -
/usr/local/Cellar/softhsm/2.1.0/lib/softhsm/libsofthsm2.so:/etc/hyperledger/fabric/libsofthsm2.so
```

7. 档案格式

1 Fabric CA 服务器的配置文件格式

在服务器的主目录中创建一个默认配置文件（有关更多信息，请参见 Fabric CA Server 部分）。以下链接显示了示例服务器配置文件。

2 Fabric CA 客户端的配置文件格式

在客户端的主目录中创建一个默认配置文件（有关更多信息，请参见 Fabric CA Client 部分）。以下链接显示了示例客户端配置文件。

8. 故障排除

1. 如果尝试执行或时在 OSX 上看到错误，则 https://github.com/golang/go/issues/19734 上有一个长线程描述此问题。简短的答案是要变通解决此问题，您可以运行以下命令：`Killed: 9fabric-ca-clientfabric-ca-server`
2. *# sudo ln -s / usr / bin / true / usr / local / bin / dsymutil*
3. 如果发生以下事件序列，将发生错误：`[ERROR] No certificates found for provided serial and aki`

    a. 您发出 *fabric-ca-client enroll* 命令，创建一个注册证书（即 ECert）。这会将 ECert 的副本存储在 fabric-ca-server 的数据库中。
    b. Fabric-ca-server 的数据库被删除并重新创建，因此从步骤 “ a” 中丢失了 ECert。例如，如果您停止并重新启动托管 Fabric-ca-server 的 Docker 容器，但是您的 fabric-ca-server 使用的是默认的 sqlite 数据库，并且该数据库文件未存储在卷上，因此不是持久性的，则可能会发生这种情况。
    c. 您发出 *Fabric-ca-client register* 命令或任何其他尝试使用步骤 “ a” 中的 ECert 的命令。在这种情况下，由于数据库不再包含 ECert，因此 将发生。
    `[ERROR] No certificates found for provided serial and aki`

    要解决此错误，您必须通过重复步骤 “ a” 重新注册。这将发出一个新的 ECert，该证书将存储在当前数据库中。
4. 当向使用共享 sqlite3 数据库的 Fabric CA Server 群集发送多个并行请求时，服务器偶尔会返回 “数据库锁定” 错误。这很可能是因为数据库事务在等待释放数据库锁（由另一个集群成员持有）时超时。这是无效的配置，因为 sqlite 是嵌入式数据库，这意味着 Fabric CA 服务器群集必须通过共享文件系统共享同一文件，这会引入 SPoF（单点故障），这与群集拓扑的目的相矛盾。最佳实践是在群集拓扑中使用 Postgres 或 MySQL 数据库。
5. 假设 使用 Fabric CA Server 颁发的注册证书时，对等方或订购者返回了与之类似的错误 。这表明 Fabric CA Server 用于颁发证书的签名 CA 证书与 用于进行授权检查的 MSP 的 *cacerts* 或 *intermediatecerts* 文件夹中的证书不匹配。
    `Failed to deserialize creator identity, err The supplied identity is not valid, Verify() returned x509: certificate signed by unknown authority`

    用于进行授权检查的 MSP 取决于发生错误时执行的操作。例如，如果您尝试在对等方上安装链码，则使用对等方文件系统上的本地 MSP；否则，将使用对等方的本地 MSP。否则，如果您正在执行某些特定于通道的操作，例如在特定通道上实例化链码，则将使用通道的创始块或最新配置块中的 MSP。

    要确认这就是问题所在，比较登记证书在证书（或多个）的 SKI（主题密钥识别）的 AKI（权威密钥标识符）*的 cacerts* 和 *intermediatecerts* 适当的 MSP 文件夹中。命令 *openssl x509 -in <PEM-file> -*

*noout -text | grep -A1 " Authority Key Identifier"（授权密钥标识符）*将显示 AKI 和 *openssl x509 -in <PEM-file> -noout -text | grep -A1 "主题密钥标识符"*将显示 SKI。如果它们不相等，则您已确认这是错误的原因。

发生这种情况可能有多种原因，其中包括：

a. 你用 *cryptogen* 生成密钥材料，但没有启动*织物 CA 服务器*与所产生的签名密钥和证书 *cryptogen*。

   解决（假设 *FABRIC_CA_SERVER_HOME* 设置为 *fabric-ca-server* 的主目录）：

a. 停止 *fabric-ca-server*。
   b. 将 *crypto-config / peerOrganizations / <orgName> / ca / * pem 复制*到 *$ FABRIC_CA_SERVER_HOME / ca-cert.pem 中*。
   c. 将 *crypto-config / peerOrganizations / <orgName> / ca / * _ sk 复制*到 *$ FABRIC_CA_SERVER_HOME / msp / keystore /中*。
   d. 启动 *fabric-ca-server*。
   e. 删除任何先前颁发的注册证书，然后通过再次注册获得新证书。
b. 生成创世块后，您删除并重新创建了 Fabric CA Server 使用的 CA 签名密钥和证书。如果 Fabric CA Server 在 Docker 容器中运行，重新启动了容器且其主目录不在卷装载中，则可能发生这种情况。在这种情况下，Fabric CA Server 将创建一个新的 CA 签名密钥和证书。

   假设您将无法恢复原来的 CA 签名密钥，从这种情况中恢复的唯一方法是，以更新证书*的 cacerts*（或 *intermediatecerts* 相应的 MSP 到新的 CA 证书）。