JavaChain平台CA节点子系统

命令行\_用户操作手册

2018-07-15

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| V3.0.0.0 | 张翀 |  | 2018.5.28 | 初始版本 |
| V3.0.0.1 | 张翀 |  | 2018.6.21 | 修改命令操作说明 |
| V3.0.0.2 | 张翀 |  | 2018.7.15 | 修改GETCAINFO命令操作说明  修改日志配置说明  修改register命令注册名的要求说明 |
| V3.0.0.3 | 张翀 |  | 2018.7.30 | 1、修改用户操作手册中getinfo章节的操作步骤，添加一步先放好配置文件ca-client-config.yaml  2、修改签发证书用户名说明  3、将lib提出来以后，修改命令中lib命令的使用 |
| V3.0.0.4 | 张翀 |  | 2018.8.02 | 1、补充说明getcainfo命令的操作步骤 |
| V3.0.0.5 | 张翀 |  | 2018.9.10 | 1、修改为符合Linux系统的描述 |

目 录

[**1** **项目描述** 5](#_Toc520989862)

[1.1 文档目标 5](#_Toc520989863)

[1.2 读者对象 5](#_Toc520989864)

[**2** **环境需求** 5](#_Toc520989865)

[2.1 部署需求 5](#_Toc520989866)

[2.2 证书和密钥的相关要求 5](#_Toc520989867)

[2.2.1 签名算法 5](#_Toc520989868)

[2.2.2 密钥存储 6](#_Toc520989869)

[2.2.3 CA返回的身份证书 6](#_Toc520989870)

[2.2.4 命令执行顺序 6](#_Toc520989871)

[**3** **使用说明** 6](#_Toc520989872)

[3.1 运行目录说明 6](#_Toc520989873)

[3.2 工作目录说明 7](#_Toc520989874)

[3.3 日志配置文件说明 8](#_Toc520989875)

[**4** **接口说明** 8](#_Toc520989876)

[4.1 命令实现类 8](#_Toc520989877)

[4.2 接口概要 9](#_Toc520989878)

[4.2.1 接口协议版本 9](#_Toc520989879)

[4.3 获取CA信息命令（cainfo） 9](#_Toc520989880)

[4.3.1 使用步骤 9](#_Toc520989881)

[4.3.2 命令 10](#_Toc520989882)

[4.3.3 请求参数 10](#_Toc520989883)

[4.3.4 返回参数 11](#_Toc520989884)

[4.4 签发身份证书命令（enroll） 11](#_Toc520989885)

[4.4.1 使用步骤 11](#_Toc520989886)

[4.4.2 命令 12](#_Toc520989887)

[4.4.3 请求参数 12](#_Toc520989888)

[4.4.4 返回参数 13](#_Toc520989889)

[4.5 重新签发证书接口（reenroll） 13](#_Toc520989890)

[4.5.1 使用步骤 13](#_Toc520989891)

[4.5.2 命令 14](#_Toc520989892)

[4.5.3 请求参数 14](#_Toc520989893)

[4.5.4 返回参数 15](#_Toc520989894)

[4.6 注册用户实体身份接口（register） 15](#_Toc520989895)

[4.6.1 使用步骤 15](#_Toc520989896)

[4.6.2 命令 16](#_Toc520989897)

[4.6.3 请求参数 16](#_Toc520989898)

[4.6.4 返回参数 17](#_Toc520989899)

[4.7 吊销证书接口（revoke） 17](#_Toc520989900)

[4.7.1 使用步骤 17](#_Toc520989901)

[4.7.2 命令 17](#_Toc520989902)

[4.7.3 请求参数 18](#_Toc520989903)

[4.7.4 返回参数 18](#_Toc520989904)

[4.8 批量查询证书（tcert） 18](#_Toc520989905)

[**5** **配置文件** 19](#_Toc520989906)

[**6** **使用Jar进行调用的代码示例** 20](#_Toc520989907)

[6.1 获取CA信息命令（cainfo） 20](#_Toc520989908)

[6.2 签发身份证书命令（enroll） 21](#_Toc520989909)

[6.3 重新签发身份证书命令（reenroll） 22](#_Toc520989910)

[6.4 注册用户实体身份命令（register） 23](#_Toc520989911)

[6.5 吊销证书命令（revoke） 23](#_Toc520989912)

# **项目描述**

该项目的主要内容是为JavaChain平台提供CA服务命令行工具，主要功能包括：申请下载，重新申请并下载以及吊销SM2 国密算法签名的证书，管理用户信息和权限，管理CA信息。

## 文档目标

CA子系统命令行工具使用说明书。

## 读者对象

供BCIA javachain的平台开发者、应用开发者及应用最终用户参考使用。

# **环境需求**

## 部署需求

JAVA环境需要Oracle JDK 1.8以上。

## 证书和密钥的相关要求

### 签名算法

必须要求是使用 SM2 国密算法签名算法来产生CSR。

### 密钥存储

按照pem格式存储密钥文件，放到指定目录（/工作目录/msp/keystore）下（详见 使用说明3.1节）。

### CA返回的身份证书

是使用 SM2 国密算法签名的证书，存储于对应目录下（详见 使用说明3.1节），以pem文件格式存储证书。

### 命令执行顺序

除了getcainfo命令之外，其他命令运行之前都必须在enroll命令之后。其他命令都必须要使用身份证书来保证安全性。

# **使用说明**

## 运行目录说明



config目录存放日志配置文件 logback.xml。

TestData目录存放样例数据。

tmp目录存放reenroll命令的临时私钥,在申请证书成功后会将该临时密钥移除,并覆盖写入对应的 keystore目录下的密钥文件中。

## 工作目录说明

用户先选定工作目录。命令行工具会在enroll命令执行后创建如下图3-1所示的目录结构：

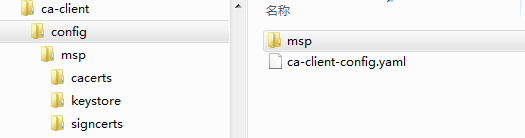


图3‑1

enroll命令执行以后,默认的配置文件ca-client-config.yaml会产生在<user.dir>/ca-client/config 目录下，在ca-client目录下会建立msp目录（<user.dir>/ca-client/msp），该目录下会有以下三个目录：

1. cacerts目录：用于存放ca的证书文件。
2. keystore目录：用于存放用户签名csr所用的私钥文件（需要用户生成后放入该目录）。
3. signcerts目录：用于存放用户enroll得到的身份证书文件（enroll命令后存入该目录）。

注意:所有文件都以pem 格式编码.请把私钥文件也按找私钥的pem格式存储。

等register成功后，会在msp目录下建立新用户的对应目录，如下图3-2所示：其中的test 目录等都是新用户目录，每个新用户目录下有自己对应的私钥和签名证书目录。

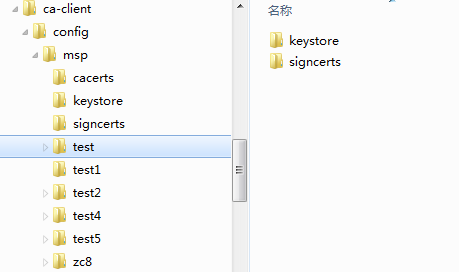


图3‑2

对应的用户注册信息数据文件(registers.dat)和证书与用户绑定关系的数据文件(enroll-id.dat)会在msp目录下。

## 日志配置文件说明

目前将日志配置文件置于config目录下，config/logback.xml，便于外部操作可配置。

# **接口说明**

## 命令实现类

如表格4-1所示：

|  |  |
| --- | --- |
| Enroll命令 | com.cfca.ra.command.internal.enroll.EnrollCommand |
| Reenroll命令 | com.cfca.ra.command.internal.reenroll. ReenrollCommand |
| Register命令 | com.cfca.ra.command.internal.register. RegisterCommand |
| Revoke命令 | com.cfca.ra.command.internal.revoke. RevokeCommand |
| Getcainfo命令 | com.cfca.ra.command.internal.getcainfo.GetCAInfoCommand |

表格4‑1

## 接口概要

用户通过命令行的方式访问ca服务器接口，如下图4-1所示：



图4‑1

Ca client负责解析前端的cmd和参数，包装成http请求发送给CA，获取CA返回的reponse后，再返回给前端。

注意，执行命令的用户名密码配置在配置文件的admin，adminpwd字段中，或者在命令行工具的json文件中指定，后者有更高的优先级。

总共6个命令，对应6个http请求。

如下表所示：其中jsonFile 是包含了Json请求的文件路径。

|  |  |
| --- | --- |
| 获取CA信息 | ca-client cainfo –h host –p port –a jsonFile |
| 签发证书 | ca-client enroll –h host –p port –a jsonFile |
| 重新签发证书 | ca-client reenroll –h host –p port –a jsonFile |
| 注册用户实体身份 | ca-client register –h host –p port –a jsonFile |
| 吊销证书 | ca-client revoke –h host –p port –a jsonFile |
| 批量查询证书 | ca-client tcert –h host –p port –a jsonFile |

### 接口协议版本

本接口协议版本号为：0.0.1。

## 获取CA信息命令（cainfo）

获取CA服务的基本信息命令，包括CA的签名根证书链P7B文件。

注意这里获取的是没有经过Base64编码的二进制p7b文件

### 使用步骤

使用api来测试可以参照以下步骤：

第一步先将配置文件ca-client-config.yaml放入ca-client/config目录下。

第二步:将请求内容按照Json格式写入cainfo.json文件，通过-a参数指定 [cainfo.json 文件路径]。cainfo.json 文件内容如下节所述。具体路径名自行定义，然后文件名路径通过-a参数指定。

第三步：在控制台执行命令。

使用命令行控制台来测试基本与上述步骤一致。

### 命令

java -Djava.ext.dirs=./lib -Dlogback.configurationFile=./config/logback.xml -jar RA-Command-xxxx.jar cainfo -h localhost -p 8089 -a TestData/cainfo.json

使用时注意指定依赖库(-Djava.ext.dirs=依赖库路径)，cainfo.json文件名以及路径可以自行指定，文件中就是json请求:

{"caName":"CFCA"}

下载下来的文件会放入ca-client/config/msp/cacerts目录下

文件名以 主机名-端口号-CA名.pem命名:

localhost-8089-CFCA.pem

### 请求参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **参数名称** | **类型** | **参数说明** | **是否可空** | **样例** |
| **业务参数** | | | | | |
| canamne | ca名称 | String | 指定ca名称 | 不可空 | CFCA |

### 返回参数

如果遇到错误会报出CommandException

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **参数名称** | **类型** | **参数说明** | **是否可空** | **样例** |
| 业务参数 | | | | | |
| Result | 处理结果 | JsonObject | {“caname”:ca名称(string), “cachain”:ca证书链文件的B64编码字符串} | 可空 |  |

## 签发身份证书命令（enroll）

接收用户信息，进行用户登记，申请身份证书的命令。该命令会读取配置文件中admin和adminpwd两项来建立鉴权信息，表明申请证书的用户。

服务器会校验该鉴权信息验证用户是否有申请权限。并且可以通过配置文件来定制化CSR部分，详细配置如下图所示。

注意：每一个用户名为了表示其唯一性，也是为了防止第三方RA限制单用户申请的证书数量，在这用户第一次使用命令行工具或者api的时候，会由内部sdk产生一个随机的10位字符串拼接在用户名后,以”-”号连接。这才是服务器最终使用的申请证书的用户名，在带回来的证书dn中有所体现。

### 使用步骤

使用api来测试可以参照以下步骤：

第一步:使用SM2国密算法生成密钥对，将生成的密钥存入<工作目录>/ca-client/config/msp/keystore 目录下,过程可以参考demo.EnrollCommandDemo中的代码(在调用enroll命令前,需要调用 final CsrResult result = CsrUtils.genCSR(algo, names);CsrUtils.storeMyPrivateKey(result, username); 存放生成好的密钥)。

第二步：需要将上一步生成的**CSR**以及其他的请求内容按照Json格式写入enroll.json 文件，通过-a参数指定 [enroll.json 文件路径]。enroll.json 文件内容如下节所述。具体路径名自行定义，然后文件名路径通过-a参数指定。

第三步：在控制台执行命令。

第四步：在对应目录（ca-client/config/msp/[enroll用户目录]/signcerts）下获得身份证书。其中enroll用户目录 ,如果用户是admin则对应的就是msp目录。

使用命令行控制台来测试基本与上述步骤一致。

### 命令

java -Djava.ext.dirs=./lib -Dlogback.configurationFile=./config/logback.xml -jar RA-Command-xxxx.jar enroll -h localhost -p 8089 -a enroll.json

使用时注意指定依赖库(-Djava.ext.dirs=依赖库路径)，enroll.json是自行指定的文件名，文件中就是json请求:

{"label":"","username":"admin","password":"1234","profile":"H09358028","csrConfig":{"cn":"C=CN,O=CFCA TEST OCA1,OU=Local RA,OU=Individual-1,CN=051@admin@ZH09358028@111","names":"CN=051@testName@Z1234567890@53,OU=Individual-3,OU=Local RA,O=CFCA TEST CA,C=CN","hosts":["<<<MYHOST>>>"],"key":{"algo":"SM2","size":256},"ca":{"pathlen":0,"pathlenzero":0,"expiry":-1}},"caName":"CFCA","request": **CSR**}

### 请求参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **参数名称** | **类型** | **参数说明** | **是否可空** | **样例** |
| **业务参数** | | | | | |
| request | 请求详细信息 | String | 构建的csr的b64字符串 | 不可空 |  |
| profile | profile信息 | String | 签名模板,暂时不传 | 可空 |  |
| label | label信息 | String | HSM操作中使用的标签 | 不可空 |  |
| caname | ca名称 | String | 请求指定的CA名称 | 不可空 |  |

### 返回参数

如果遇到错误会报出CommandException

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **参数名称** | **类型** | **参数说明** | **是否可空** | **样例** |
| **业务参数** | | | | | |
| Result | 处理结果 | String | Base64编码格式的身份证书, Success为true时,会有值 | 可空 | 样例太长,自行生成 |
| Success | 成功标识 | boolean | 布尔值，指示请求是否成功 | 不可空 | true |
| Errors | 处理错误 | array | 每个元素是一个对象{ code:xxxxx,message:xxxx}  Success为false时,会有值 | 可空 | 一组错误消息（即代码和字符串消息） |
| Messages | 处理信息 | array | 每个元素是一个对象{ code:xxxxx,message:xxxx} | 可空 | 一系列信息性消息（即代码和字符串消息） |

## 重新签发证书接口（reenroll）

当身份证书过期后，调用该命令重新进行用户登记，申请身份证书的命令。该命令会使用已有的身份证书进行签名，建立鉴权信息，服务器会校验该鉴权信息。

### 使用步骤

使用api来测试可以参照以下步骤：

第一步:使用SM2国密算法生成密钥对，将生成的密钥存入<新密钥文件路径>, 注意新密钥不能覆盖enroll命令生成的老密钥，因为老密钥要用于生成签名。具体生产密钥对代码可以参考Demo。在命令结束后，命令行工具自身会将老密钥替换成新密钥。生成的新密钥路径通过-key 命令行参数指定新密钥路径。

第二步：需要将上一步生成的**CSR**以及其他的请求内容按照Json格式写入reenroll.json 文件，通过-a参数指定 [reenroll.json 文件路径]。reenroll.json 文件内容跟上节一致。具体路径名自行定义，然后文件名路径通过-a参数指定。

第三步：调用API执行命令。

第四步：在对应目录（ca-client/config/msp/[reenroll用户目录]/signcerts）下获得身份证书。其中reenroll用户目录 ,如果用户是admin则对应的就是msp目录。

使用命令行控制台来测试基本与上述步骤一致，注意一点,调用reenrol命令控制台时，所使用的身份证书的DN必须与ca-client/config/ ca-client-config.yaml配置文件中的字段Csr.cn一致。

### 命令

java -Djava.ext.dirs=./lib -Dlogback.configurationFile=./config/logback.xml -jar RA-Command-xxxx.jar reenroll -h localhost -p 8089 -a reenroll.json -key <newKeyFile>

使用时注意指定依赖库(-Djava.ext.dirs=依赖库路径)，reenroll.json是指定的文件名，文件中就是json请求:

{"label":"","username":"admin","password":"1234","profile":"H09358028","csrConfig":{"cn":"C=CN,O=CFCA TEST OCA1,OU=Local RA,OU=Individual-1,CN=051@admin@ZH09358028@111","names":"CN=051@testName@Z1234567890@53,OU=Individual-3,OU=Local RA,O=CFCA TEST CA,C=CN","hosts":["<<<MYHOST>>>"],"key":{"algo":"SM2","size":256},"ca":{"pathlen":0,"pathlenzero":0,"expiry":-1}},"caName":"CFCA","request":**CSR**}

### 请求参数

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **参数** | **参数名称** | **类型** | | **参数说明** | | **是否可空** | **样例** |
| **业务参数** | | | | | | | |
| request | 请求详细信息 | | String | | 构建的csr的b64字符串 | 不可空 |  |
| profile | profil信息 | | String | | 签名模板,暂时不可用 | 可空 |  |
| label | label信息 | | String | | HSM操作中使用的标签 | 不可空 |  |
| caname | ca名称 | | String | | 请求指定的CA名称 | 不可空 |  |

### 返回参数

如果遇到错误会报出CommandException。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **参数名称** | **类型** | **参数说明** | **是否可空** | **样例** |
| **业务参数** | | | | | |
| Result | 处理结果 | JsonObject | {“cert”:Base64编码格式的身份证书} | 可空 |  |

## 注册用户实体身份接口（register）

接收用户信息，进行新用户注册命令。注册时要使用身份证书建立鉴权信息（token），注册成功后，会得到新注册用户的密码。

注意：注册用户名长度限定在[3,64]字节，用户名包含特殊字符时注册失败。

### 使用步骤

使用api来测试可以参照以下步骤：

第一步: 请求内容按照Json格式写入register.json 文件。具体路径名自行定义，然后文件名路径通过-a参数指定。

第二步：在控制台执行命令。

第三步：返回新用户密码。

使用命令行控制台来测试基本与上述步骤一致。

### 命令

java -Djava.ext.dirs=./lib -Dlogback.configurationFile=./config/logback.xml -jar RA-Command-xxxx.jar register -h localhost -p 8089 -a register.json

使用时注意指定依赖库(-Djava.ext.dirs=依赖库路径)，register.json是指定的文件名，文件中就是json请求:

{"name":"zc10","type":"user","secret":"1234","maxEnrollments":2,"affiliation":"org.department.c","attributes":[{"name":"hf.Revoker","value":"true","eCert":false},{"name":"hf.Registrar.Roles","value":"client,user,peer,validator,auditor","eCert":false}],"caName":"CFCA"}

### 请求参数

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **参数** | **参数名称** | | **类型** | **参数说明** | **是否可空** | **样例** |
| **业务参数** | | | | | | |
| id | | id | String | 用户id | 不可空 |  |
| type | | 类型 | String | 类型 | 不可空 |  |
| secret | | 密码 | String | 密码 | 不可空 |  |
| max\_enrollments | | 数量 | Integer | 最大证书数量 | 不可空 |  |
| affiliation\_path | | 路径 | String | 路径 | 不可空 |  |
| attrs | | 注册属性 | Json | "attrs": [  {  "name": "string",  "value": "string"  }  ],用数组传入多个属性值 | 不可空 |  |
| caname | | ca名称 | String | ca名称 | 不可空 |  |

### 返回参数

如果遇到错误会报出CommandException。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **参数名称** | **类型** | **参数说明** | **是否可空** | **样例** |
| **业务参数** | | | | | |
| result | 处理结果 | Json | {"credentials": "string"} | 不可空 |  |

## 吊销证书接口（revoke）

吊销证书命令。通过传入的证书序列号进行吊销。

### 使用步骤

使用api来测试可以参照以下步骤：

第一步: 请求内容按照Json格式写入revoke.json 文件。具体路径名自行定义，然后文件名路径通过-a参数指定。

第二步：在控制台执行命令。

使用命令行控制台来测试基本与上述步骤一致。

### 命令

java -Dlogback.configurationFile=./config/logback.xml -Djava.ext.dirs=./lib -jar RA-Command-xxxx.jar revoke -h localhost -p 8089 -a revoke.json

使用时注意指定依赖库(-Djava.ext.dirs=依赖库路径)，revoke.json是指定的文件名，文件中就是json请求:

{"id":"admin","aki":"ssss","serial":"1032940241","reason":"expire","caname":"CFCA"}

### 请求参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **参数名称** | **类型** | **参数说明** | **是否可空** | **样例** |
| **业务参数** | | | | | |
| id | Id | String | 用户名 | 不可空 | test |
| aki | Aki | String | 要撤销的证书的授权密钥标识符（Authority Key Identifier） | 不可空 | “” |
| serial | Serial | String | 证书序列号 | 不可空 | 1032137068 |
| reason | Reason | String | 吊销原因 | 不可空 | expire |
| caname | Caname | String | Ca名称 | 不可空 | CFCA |

### 返回参数

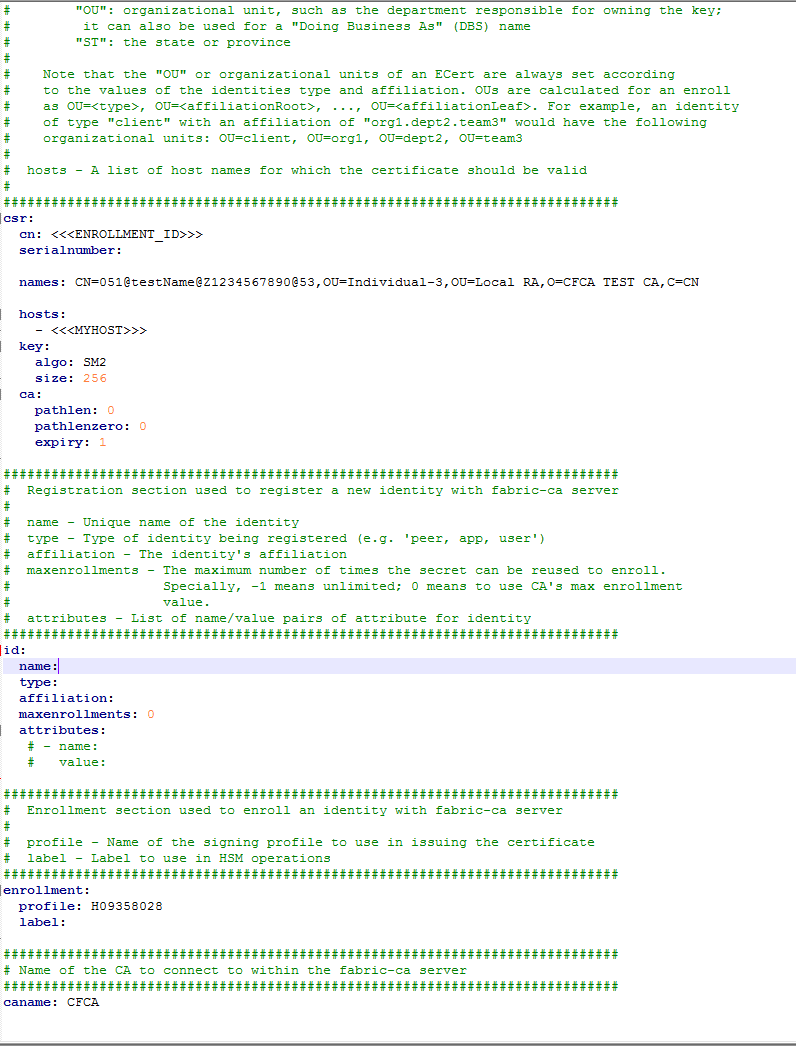
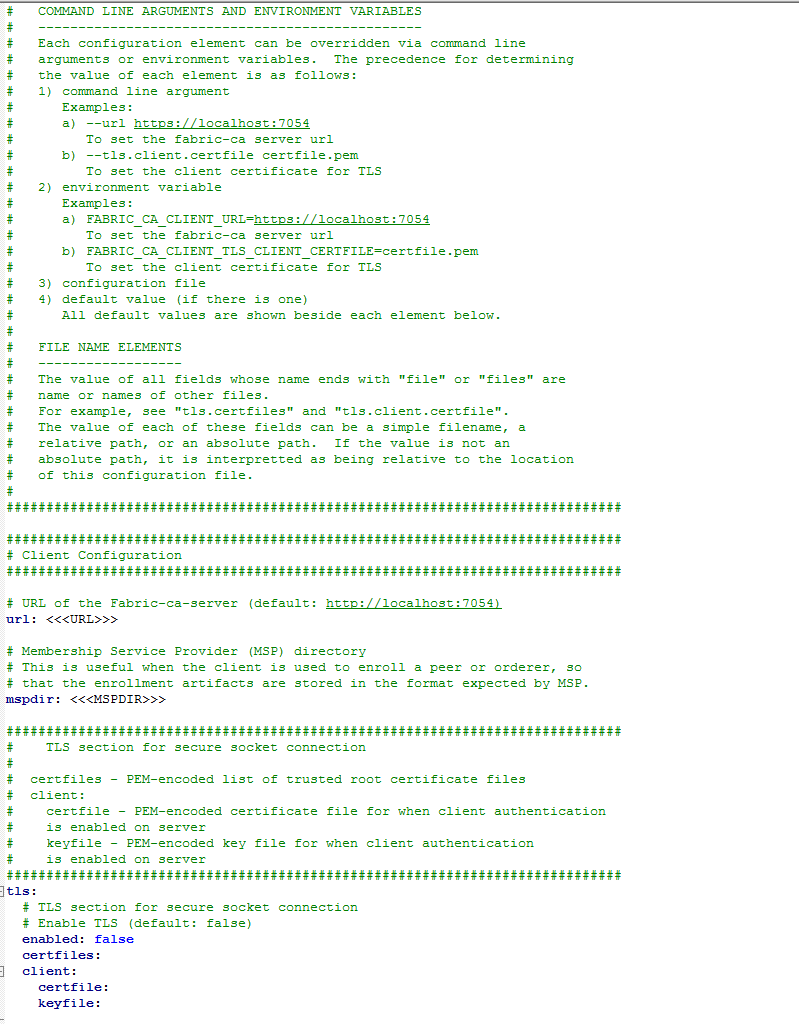
如果遇到错误会报出CommandException。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **参数名称** | **类型** | **参数说明** | **是否可空** | **样例** |
| **业务参数** | | | | | |
| result | 处理结果 | Json | {"result": "string"} | 不可空 |  |

## 批量查询证书（tcert）

批量查询证书命令。该版本暂不支持。

# **配置文件**



# **使用Jar进行调用的代码示例**

## 获取CA信息命令（cainfo）

public static void main(String[] args) throws Exception {

final GetCAInfoRequest caInfoRequest = new GetCAInfoRequest("CFCA");  
final Gson gson = new GsonBuilder().disableHtmlEscaping().create();  
final String jsonFile = "TestData/cainfo.json";  
final String request = gson.toJson(caInfoRequest);  
System.out.println("request=" + request);  
FileUtils.writeStringToFile(new File(jsonFile), request);  
  
final GetCAInfoCommand getCAInfoCommand = new GetCAInfoCommand();  
getCAInfoCommand.prepare(new String[] { "cainfo", "-h", "localhost", "-p", "8089", "-a", jsonFile });  
final JsonObject result = getCAInfoCommand.execute();  
System.out.println(result);

}

## 签发身份证书命令（enroll）

private static ConfigBean loadConfigFile() throws CommandException {  
 try {  
 return ConfigUtils.load("ca-client/config/ca-client-config.yaml");  
 } catch (Exception e) {  
 throw new CommandException(CommandException.REASON\_CODE\_ENROLL\_COMMAND\_LOAD\_CONFIG\_FAILED, e);  
 }  
}  
  
public static void main(String[] args) throws Exception {  
 final ConfigBean configBean = loadConfigFile();  
 String profile = configBean.getEnrollment().getProfile();  
 CsrConfig csrConfig = configBean.getCsr();  
 String caName = configBean.getCaname();  
 final String username = "admin";  
 final String password = "1234";  
  
 final String algo = csrConfig.getKey().getAlgo();  
 final String names = csrConfig.getNames();  
 final CsrResult result = CsrUtils.genCSR(algo, names);  
 final String csr = result.getCsr();  
 System.out.println("Csr=" + csr);  
 CsrUtils.storeMyPrivateKey(result, username);  
  
 final EnrollmentRequest.Builder builder = new EnrollmentRequest.Builder(csr, username, password, profile, csrConfig, caName);  
 final EnrollmentRequest enrollmentRequest = builder.build();  
 final Gson gson = new GsonBuilder().disableHtmlEscaping().create();  
 final String jsonFile = "TestData/enroll.json";  
 FileUtils.writeStringToFile(new File(jsonFile), gson.toJson(enrollmentRequest));  
  
 final EnrollCommand enrollCommand = new EnrollCommand();  
 String[] args1 = new String[]{"enroll", "-h", "localhost", "-p", "8089", "-a", jsonFile};  
 enrollCommand.prepare(args1);  
 final JsonObject response = enrollCommand.execute();  
 System.out.println(response);  
}

## 重新签发身份证书命令（reenroll）

private void testReenroll() throws Exception {

final ConfigBean configBean = loadConfigFile();

String profile = configBean.getEnrollment().getProfile();

CsrConfig csrConfig = configBean.getCsr();

String caName = configBean.getCaname();

final String algo = csrConfig.getKey().getAlgo();

final String names = csrConfig.getNames();

final CsrResult result = CsrUtils.genCSR(algo, names);

final String keyDir = "D:\\R15\\P1552\\dev\\blockchain\\command\\ca-client\\config\\msp\\tmp\\keystore";

final String keyFile = "key.pem";

CsrUtils.storePrivateKey(result, keyDir, keyFile);

final String csr = result.getCsr();

System.out.println("CSR=" + csr);

final String username = "admin";

final String password = "1234";

final ReenrollmentRequest.Builder builder = new ReenrollmentRequest.Builder(csr, username, password, profile, csrConfig, caName);

final ReenrollmentRequest reenrollmentRequest = builder.build();

final Gson gson = new GsonBuilder().disableHtmlEscaping().create();

final String jsonFile = "TestData/reenroll.json";

final String request = gson.toJson(reenrollmentRequest);

System.out.println("request=" + request);

FileUtils.writeStringToFile(new File(jsonFile), request);

final String keyFilePath = String.join(File.separator, keyDir, keyFile);

String[] args = new String[]{"reenroll", "-h", "localhost", "-p", "8089", "-a", jsonFile, "-key", keyFilePath};

ReenrollCommand reenrollCommand = new ReenrollCommand();

reenrollCommand.prepare(args);

reenrollCommand.execute();

}

## 注册用户实体身份命令（register）

private void testRegister() throws Exception {

final RegistrationRequest.Builder builder = new RegistrationRequest.Builder();

final ArrayList<UserAttrs> v = new ArrayList<>();

v.add(new UserAttrs("hf.Revoker", "true"));

v.add(new UserAttrs("hf.Registrar.Roles", "client,user,peer,validator,auditor"));

builder.name("zc10").caName("CFCA").affiliation("org.department.c").attributes(v).maxEnrollments(2).secret("1234").type("user");

final RegistrationRequest registrationRequest = builder.build();

final String jsonFile = "TestData/register.json";

final Gson gson = new GsonBuilder().disableHtmlEscaping().create();

final String request = gson.toJson(registrationRequest);

System.out.println("request=" + request);

FileUtils.writeStringToFile(new File(jsonFile), request);

final RegisterCommand registerCommand = new RegisterCommand();

String[] args = new String[]{"register", "-h", "localhost", "-p", "8089", "-a", jsonFile};

registerCommand.prepare(args);

registerCommand.execute();

}

## 吊销证书命令（revoke）

private void testRevoke() throws Exception{

String id = "admin";

String aki = "ssss";

String serial = "1032940241";

String reason = "expire";

String caname = "CFCA";

final RevokeRequest revokeRequest = new RevokeRequest(id, aki, serial, reason, caname);

final String jsonFile = "TestData/revoke.json";

final Gson gson = new GsonBuilder().disableHtmlEscaping().create();

final String request = gson.toJson(revokeRequest);

System.out.println("request=" + request);

FileUtils.writeStringToFile(new File(jsonFile), request);

String[] args = new String[]{"revoke", "-h", "localhost", "-p", "8089", "-a", jsonFile};

RevokeCommand revokeCommand = new RevokeCommand();

revokeCommand.prepare(args);

final JsonObject result = revokeCommand.execute();

System.out.println(result);

}