JavaChain平台CA节点子系统

命令行\_用户操作手册

2018-05-28

版 本 历 史

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 版本/状态 | 作者 | 参与者 | 起止日期 | 备注 |
| V3.0.0 | 张翀 |  | 2018.5.28 | 初始版本 |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

目 录

[**1** **项目描述** 5](#_Toc515616935)

[1.1 文档目标 5](#_Toc515616936)

[1.2 读者对象 5](#_Toc515616937)

[**2** **环境需求** 5](#_Toc515616938)

[2.1 部署需求 5](#_Toc515616939)

[2.2 证书和密钥的相关要求 5](#_Toc515616940)

[2.2.1 签名算法 5](#_Toc515616941)

[2.2.2 密钥存储 6](#_Toc515616942)

[2.2.3 CA返回的身份证书 6](#_Toc515616943)

[**3** **使用说明** 6](#_Toc515616944)

[3.1 工具目录说明 6](#_Toc515616945)

[**4** **接口说明** 7](#_Toc515616946)

[4.1 命令实现类 7](#_Toc515616947)

[4.2 接口概要 8](#_Toc515616948)

[4.2.1 接口协议版本 8](#_Toc515616949)

[4.3 接口公共参数 8](#_Toc515616950)

[4.3.1 接口请求公共参数 8](#_Toc515616951)

[4.4 获取CA信息命令（cainfo） 9](#_Toc515616952)

[4.4.1 命令 9](#_Toc515616953)

[4.4.2 请求参数 9](#_Toc515616954)

[4.4.3 返回参数 9](#_Toc515616955)

[4.5 签发身份证书命令（enroll） 10](#_Toc515616956)

[4.5.1 命令 10](#_Toc515616957)

[4.5.2 请求参数 10](#_Toc515616958)

[4.5.3 返回参数 10](#_Toc515616959)

[4.6 重新签发证书接口（reenroll） 11](#_Toc515616960)

[4.6.1 命令 11](#_Toc515616961)

[4.6.2 请求参数 11](#_Toc515616962)

[4.6.3 返回参数 11](#_Toc515616963)

[4.7 注册用户实体身份接口（register） 12](#_Toc515616964)

[4.7.1 命令 12](#_Toc515616965)

[4.7.2 请求参数 12](#_Toc515616966)

[4.7.3 返回参数 13](#_Toc515616967)

[4.8 吊销证书接口（revoke） 13](#_Toc515616968)

[4.8.1 命令 13](#_Toc515616969)

[4.8.2 请求参数 13](#_Toc515616970)

[4.8.3 返回参数 13](#_Toc515616971)

[4.9 批量查询证书（tcert） 14](#_Toc515616972)

[**5** **配置文件** 14](#_Toc515616973)

[**6** **CA客户端命令示例** 17](#_Toc515616974)

[**7** **代码示例** 17](#_Toc515616975)

[7.1 获取CA信息命令（cainfo） 17](#_Toc515616976)

[7.2 签发身份证书命令（enroll） 17](#_Toc515616977)

[7.3 重新签发身份证书命令（reenroll） 18](#_Toc515616978)

[7.4 注册用户实体身份命令（register） 19](#_Toc515616979)

[7.5 吊销证书命令（revoke） 19](#_Toc515616980)

[**8** **错误代码** 20](#_Toc515616981)

[8.1 错误代码表 20](#_Toc515616982)

# **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。

# **使用说明**

## 工具目录说明

用户先选定工作目录,也就是程序运行的当前目录,然后请把用于签名CSR的私钥放入< user.dir >\ca-client\config\msp\keystore\下.

命令行工具运行起来以后,会在enroll命令执行后创建如下图3-1所示的目录结构：

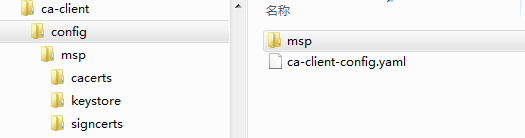


图3‑1

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。
3. signcerts目录：用于存放用户enroll得到的身份证书文件。

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。

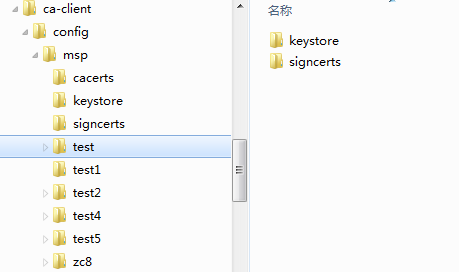


图3‑2

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。

另外用户也可以在<user.dir>目录下建立ca-client目录，放入自定义的配置文件。

# **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。

总共6个命令，对应6个http请求

如下表所示：

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

### 接口协议版本

本接口协议版本号为：0.0.1。

## 接口公共参数

### 接口请求公共参数

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **参数** | **参数名称** | **类型** | **参数说明** | **是否可空** | **样例** |
| **基本参数** | | | | | |
| --caname | ca名称 | String | Ca实例名称 | 不可空 | CFCA |
| **扩展参数** | | | | | |
| -c,--config | 配置文件路径 | String | 配置文件路径 | 可为空 | D:\config |
| -d,--debug | Debug模式 | String | Debug模式，输出更多日志内容 | 可为空 |  |
| -u,--url | ca服务器地址 | String | 进行连接的ca-server服务地址 | 可为空 | http://localhost:8089 |
| -M,--mspdir | MSP路径 | String | 生成证书存放目录MSP的路径，默认为“msp” | 可为空 | D:\msp |

## 获取CA信息命令（cainfo）

获取CA服务的基本信息命令，包括CA的签名证书或者根证书链文件。

### 命令

ca-client cainfo

### 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，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。

### 命令

[ca-client enroll](http://IP:Port/enroll)

### 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 | 不可空 |  |
| 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。

### 命令

[ca-client reenroll](http://IP:Port/reenroll)

### 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 | 不可空 |  |
| 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，同时在数据库文件中建立用户信息，包括用户与密码的绑定关系等。并且此时可以用新注册的用户名和密码调用enroll命令进行申请新的身份证书，申请成功后对应的签名证书和私钥就会存放在新用户目录下。

### 命令

[ca-client register](http://IP:Port/register)

### 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。

### 命令

ca-client revoke

### 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

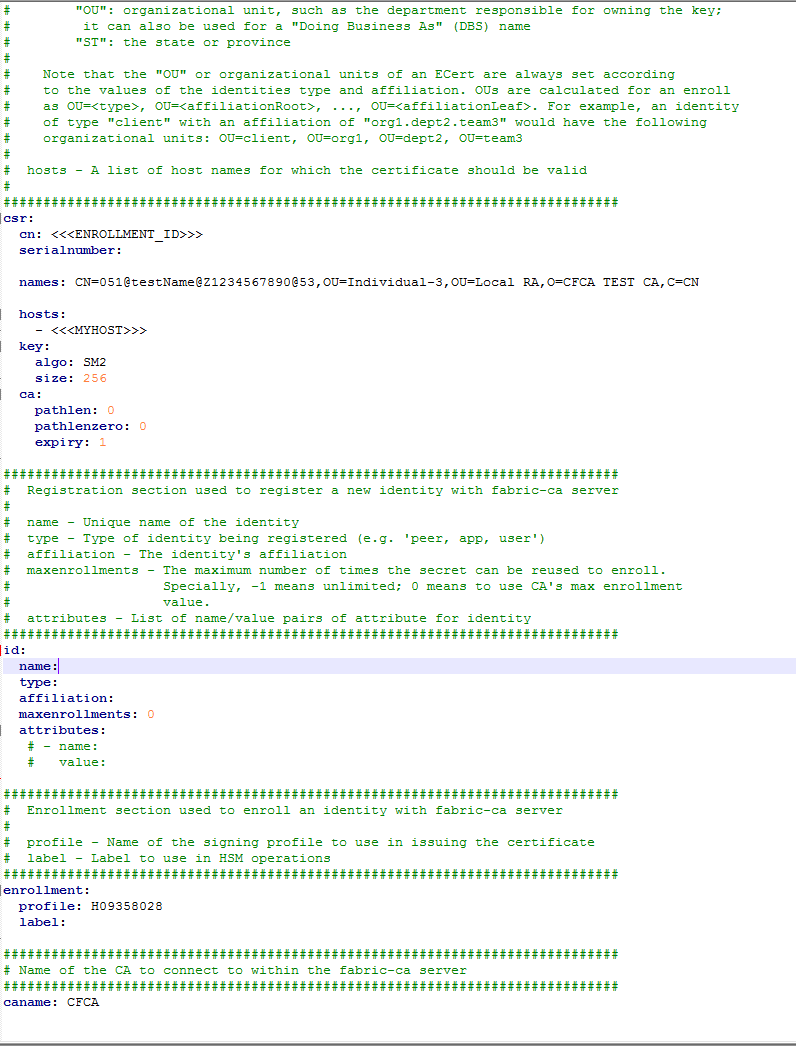
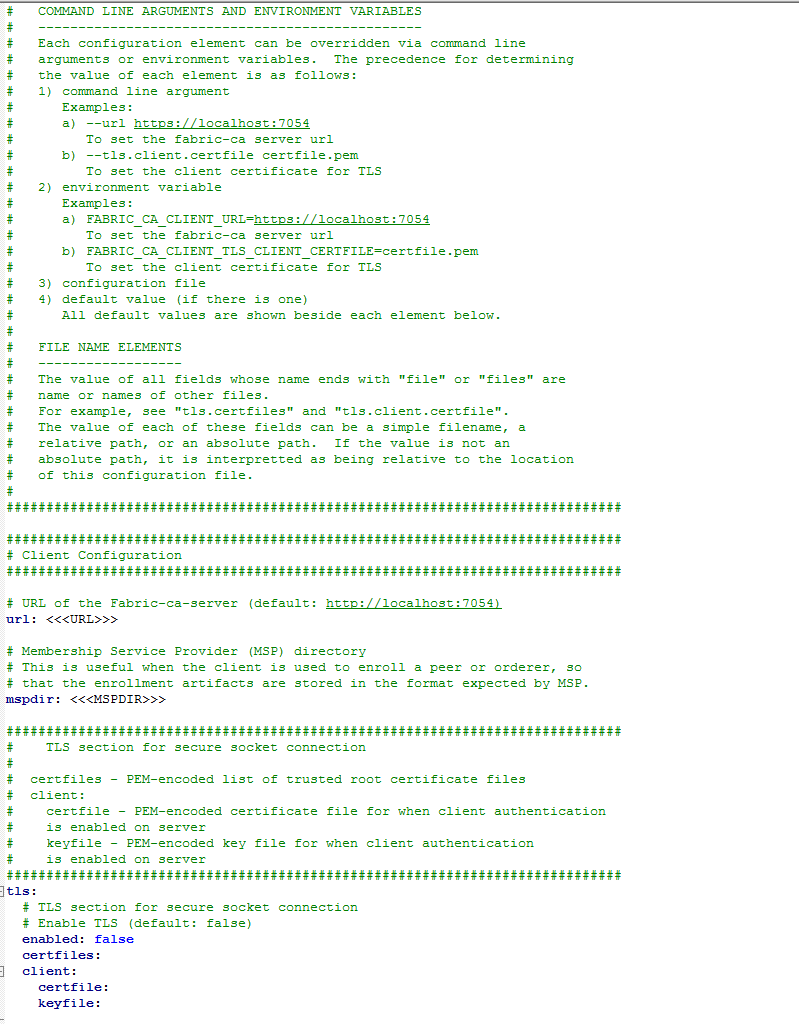
如果遇到错误会报出CommandException。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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**



# **CA客户端命令示例**

ca-client reenroll –h “host” –p “port” –a “{'request': 'requestExample','profile': 'profileExample','label': 'labelExample','caname': 'canameExample'}”

# **代码示例**

## 获取CA信息命令（cainfo）

public void testExecute() throws Exception {

final GetCAInfoCommand getCAInfoCommand = new GetCAInfoCommand();

getCAInfoCommand.prepare(new String[]{"cainfo", "-h", "localhost", "-p", "8089", "-a", "{\"caName\":\"CFCA\"}"});

final JsonObject result = getCAInfoCommand.execute();

}

## 签发身份证书命令（enroll）

private 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 CommandException{

final EnrollCommandDemo enrollCommandDemo = new EnrollCommandDemo();

enrollCommandDemo.work();

}

private void work() throws CommandException {

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);

storeMyPrivateKey(result);

final EnrollmentRequest.Builder builder = new EnrollmentRequest.Builder(result.getCsr(), "admin", "1234", profile, csrConfig, caName);

final EnrollmentRequest enrollmentRequest = builder.build();

final EnrollCommand enrollCommand = new EnrollCommand();

String[] args = new String[]{"enroll", "-h", "localhost", "-p", "8089", "-a", new Gson().toJson(enrollmentRequest)};

enrollCommand.prepare(args);

final JsonObject response = enrollCommand.execute();

System.out.println(response);

}

## 重新签发身份证书命令（reenroll）

private 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 ReenrollCommandDemo reenrollCommandDemo = new ReenrollCommandDemo();

reenrollCommandDemo.testReenroll();

}

private void testReenroll() throws Exception{

final ConfigBean configBean = loadConfigFile();

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

CsrConfig csrConfig = configBean.getCsr();

String caName = configBean.getCaname();

//"test", "dGVzdDoxMjM0"// "test2":"dGVzdDI6MTIzNA=="

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

final String names = csrConfig.getNames();

final CsrResult result = genCSR(algo, names);

storeMyPrivateKey(result);

final ReenrollmentRequest.Builder builder = new ReenrollmentRequest.Builder(result.getCsr(), "test4","dGVzdDQ6MTIzNA==", profile, csrConfig, caName);

final ReenrollmentRequest reenrollmentRequest = builder.build();

String[] args = new String[]{"enroll", "-h", "localhost", "-p", "8089", "-a", new Gson().toJson(reenrollmentRequest)};

ReenrollCommand reenrollCommand=new ReenrollCommand();

reenrollCommand.prepare(args);

reenrollCommand.execute();

}

## 注册用户实体身份命令（register）

public 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("test5").caName("CFCA").affiliation("org.department.c").attributes(v).maxEnrollments(2).secret("1234").type("user");

final RegistrationRequest registrationRequest = builder.build();

final RegisterCommand registerCommand = new RegisterCommand();

String[] args = new String[]{"register", "-h", "localhost", "-p", "8089", "-a", new Gson().toJson(registrationRequest)};

registerCommand.prepare(args);

final JsonObject result = registerCommand.execute();

}

## 吊销证书命令（revoke）

public void testRevoke() throws Exception{

String id = "admin";

String aki = "sss";

String serial = "1032162646";

String reason = "expire";

String caname = "CFCA";

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

String[] args = new String[]{"revoke", "-h", "localhost", "-p", "8089", "-a", new Gson().toJson(revokeRequest)};

revokeCommand.prepare(args);

final JsonObject result = revokeCommand.execute();

}

# **错误代码**

## 错误代码表

|  |  |
| --- | --- |
| 错误码 | 错误信息含义 |
| 0x00 | 客户端异常 |
| 0x1006 | 配置文件没有 CA 名称 |
| 0x1007 | 配置文件没有指定enroll命令的profile |
| 0x1008 | 配置文件没有指定enroll命令 |
| 0x2001 | 命令行缺少host |
| 0x2002 | 命令行缺少port |
| 0x2003 | 命令行缺少content |
| 0x2004 | 存储cachain文件失败 |
| 0x2005 | 加载配置文件失败 |
| 0x5001 | Enroll命令参数无效 |
| 0x5002 | Enroll命令网络通信失败 |
| 0x5003 | Enroll命令加载配置文件失败 |
| 0x5004 | Enroll命令更新配置文件失败 |
| 0x5005 | Enroll命令加载用户与证书绑定关系失败 |
| 0x5006 | Enroll命令更新用户与证书绑定关系失败 |
| 0x6001 | Getcainfo命令网络通信失败 |
| 0x6002 | Getcainfo命令初始化失败 |
| 0x6003 | Getcainfo命令解析命令行参数失败 |
| 0x6004 | Getcainfo命令加载配置文件失败 |
| 0x7002 | REENROLL 命令加载配置文件失败 |
| 0x7003 | REENROLL 命令网络通信失败 |
| 0x7004 | REENROLL 命令解析命令行参数失败 |
| 0x7005 | REENROLL 命令解析配置文件缺失ENROLLMENT |
| 0x8001 | REGISTER 命令解析命令行参数失败 |
| 0x8002 | REGISTER 命令网络通信失败 |
| 0x8003 | REGISTER 命令服务器返回失败的回复 |
| 0x8005 | REGISTER 命令加载注册数据库文件失败 |
| 0x8006 | REGISTER 命令更新注册数据库文件失败 |
| 0x9001 | REVOKE 命令解析命令行参数失败 |
| 0x9002 | REVOKE 命令网络通信失败 |
| 0xa001 | GETTCERT 命令网络返回失败 |
| 0xa002 | GETTCERT 命令解析命令行参数失败 |
| 0xb002 | 内部客户端load identity失败 |
| 0xb003 | 内部客户端创建 cert file 文件路径失败 |
| 0xb004 | 内部客户端创建 private key file 文件路径失败 |
| 0xb005 | 内部客户端存储 private key file 文件失败 |
| 0xb006 | 内部客户端存储 identity 失败 |
| 0xb007 | 内部客户端生成 SM2 CSR 失败 |
| 0xb008 | 内部客户端执行 enroll 失败,csr配置问题 |
| 0xb009 | 内部客户端存储 serverinfo 失败 |
| 0xb00a | 内部客户端创建 identity 失败 |
| 0xb00b | 内部客户端创建 reenrollment 网络请求失败 |
| 0xb00c | 内部客户端创建鉴权信息失败 |
| 0xb00d | 内部客户端创建CSR失败 |
| 0xb00e | 内部客户端创建失败 |
| 0xb00f | 内部客户端检查是否签发过证书失败 |
| 0xb010 | 内部客户端 reenroll 失败 |
| 0xb012 | 内部客户端 register 后服务器回复空的密码 |
| 0xc001 | 文件操作失败,检查文件是否存在失败 |
| 0xc002 | 文件操作失败,创建新文件失败 |
| 0xc003 | 文件操作失败,获取文件目录失败 |
| 0xd001 | IDENTITY 创建签名鉴权失败 |