

移动数字证书(CA)互认技术标准

CA 互认 APP 接入指南

2024 年 7 月

修订说明

版本	内容	日期
20240812	(1) 1.3 更新图 1 服务流程图及内容。 (2) 2.3.2.1 二维码内容解析接口 serviceCode 改为 aShareComponertsCode。 (3) 所有接口有 caOrgType 字段的名称调整为 caOrgCode。 (4) 调整了请求和响应示例。	20240812
20240814	(1) 4.2 附录二新增 CA 机构编码。	20240814
20240822	(1) 2.2.3.1 功能接口编码 setReceiveUrl 调整为 batchGetOpenCatalogAndComponentsInfo	20240822
20240924	(1) 2.3.9.1 推送单位用户加密证书接口，添加请求参数 orgName 单位名称	20240924

目 录

1. 接入要求	1 -
1.1. CA 互认 APP 入网要求	3 -
1.2. 电子招标投标交易系统/交易工具运营机构部署要求	4 -
2. CA 互认 APP 标准接口对接说明	9 -
2.1. 对接准备	9 -
2.2. CA 互认 APP-SDK 对接说明	9 -
2.3. CA 互认 APP 与 CA 互认共享组件对接	28 -
3. 自主联调测试进入 CA 互认共享目录	103 -
4. 附录	103 -
4.1. 附录 1:接口间调用数据加密解密约定	103 -
4.2. 附录 2:CA 机构编码	113 -

根据《移动数字证书(CA)互认技术标准》，CA 互认 APP 是依托全国工程建设招标投标领域电子招标投标公共服务平台网络体系（简称全国电子招标投标公共服务平台网络体系）实现移动 CA 数字证书跨区域、跨系统兼容互认共享的载体。为方便 CA 互认 APP 接入 CA 互认共享组件，特制订本指南。

本指南是《移动数字证书(CA)互认技术标准》的重要组成部分。

1.接入要求

全国工程建设招标投标领域电子招标投标公共服务平台网络体系面向符合本标准规范的各类招标投标监督系统、招标投标公共服务平台、电子招标投标交易系统和交易工具，以及 CA 互认 APP、CA 数字证书系统、印章机构系统等各类应用节点开放，提供开放互联和互认共享应用服务。各类应用节点应当统一遵循本标准规范要求，确保全国工程建设招标投标领域电子招标投标公共服务平台网络体系稳定运行。

1.1.全国电子招标投标公共服务平台网络体系建设运行要求

全国电子招标投标公共服务平台网络体系作为工程建设招标投标领域实现 CA 数字证书跨区域、跨系统兼容共享的载体，具有公共服务性质。其建设和运行应该符合以下要求：

（1）中国招标投标公共服务平台（以下简称“国家公共服务平台”）负责全国电子招标投标公共服务平台网络体系的组

织建设和运行监测。

（2）国家公共服务平台负责依据本标准规范发布提供全国统一的 CA 互认共享组件，为各类应用节点接入全国电子招标投标公共服务平台网络体系提供便利。CA 互认共享组件应符合“技术中立、兼容开放、源码公开、使用免费”的原则。

（3）国家公共服务平台负责公开发布符合标准的 CA 互认共享组件、共享登录注册组件、电子签名验签组件、电子签章验章组件、加密解密组件源代码，并通过必要的电子签名等技术确保 CA 互认共享组件的安全下载、应用。国家公共服务平台负责所提供的 CA 互认共享组件软件升级和漏洞修复工作。

（4）国家公共服务平台负责电子招标投标交易系统/交易工具侧 CA 互认共享组件及其基础软件的在资源服务器中的部署与维护。

（5）国家公共服务平台负责提供 CA 互认共享组件联调测试环境，供各类应用节点的运营机构自主接入和联调测试。国家公共服务平台可提供必要的使用培训指导。

（6）国家公共服务平台负责汇集公布全国范围内通过自主联调测试的电子招标投标监督系统、电子招标投标交易系统/交易工具、CA 互认 APP、CA 机构和印章机构的基本信息和自主测试报告，形成全国移动 CA 数字证书互认共享清单目录（简称“CA 互认共享目录”）。

(7) 国家公共服务平台负责汇集全国 CA 互认共享组件运行使用情况和移动 CA 数字证书互认使用情况。

(8) 国家公共服务平台应在电子招标投标交易系统/交易工具运营机构配合下，及时解决 CA 互认共享组件及其基础软件运行过程中的异常预警，保障全国电子招标投标公共服务平台网络体系的稳定运行。

1.2.CA 互认 APP 入网要求

(1) CA 互认 APP 及其入驻 CA 数字证书应当按照本技术标准要求进行研发和运营。

(2) CA 互认 APP 运营机构应当选择符合本标准和开源代码的 CA 互认共享组件接入，自主完成与 CA 互认共享组件的联调测试，联调测试通过并提交自主测试报告后，方可进入 CA 互认共享目录。

(3) CA 互认 APP 运营机构应在需要开通对接的电子招标投标交易系统/交易工具运营机构组织下与私有化部署的 CA 互认组件开展联调测试，并与电子招标投标交易系统/交易工具运营机构签署技术服务协议。CA 互认 APP 运营机构应配备专业的客服团队，及时解决用户使用过程中出现的问题。

(4) CA 互认 APP 应遵守接入的 CA 机构公布的电子认证业务规则进行业务操作。定期接受 CA 机构针对 CA 数字证书生命周期管理的审计。协助 CA 机构完成工业和信息化部、国

家密码管理局定期对 CA 机构的定期检查。

1.3.电子招标投标交易系统/交易工具运营机构部署要求

电子招标投标交易系统/交易工具运营机构是互认的移动 CA 数字证书应用的主体责任单位。

(1) 电子交易系统/交易工具运营机构应当依据本标准，细化制定自身技术验证测试标准、方式和时间（建议 15 个自然日），并在国家公共服务平台、省级招标投标公共服务平台以及自身网站予以公布。

(2) 电子招标投标交易系统/交易工具运营机构配合国家公共服务平台完成 CA 互认共享组件及其基础软件的部署，自主完成相关联调测试，提交自主测试报告，并与国家公共服务平台和其他自愿开发 CA 互认共享组件的机构签署技术服务协议。

(3) 电子招标投标交易系统/交易工具运营机构负责实现电子招标投标交易系统/交易工具、CA 互认 APP 和 CA 数字证书、电子签章软件的安全匹配与共享应用。电子招标投标交易系统/交易工具运营机构应从全国 CA 互认共享目录中自主选择 CA 互认 APP 运营机构、CA 机构和印章机构，并负责组织相关机构开展联调测试，并于 15 个自然日内完成。电子招标投标交易系统/交易工具上应当开通 3 个以上 CA 互认 APP 和 30 个以上移动 CA 数字证书。电子招标投标交易系统/交易工具运营

机构应及时与开通应用的 CA 互认 APP 运营机构、CA 机构、印章机构签订技术服务协议。

（4）电子招标投标交易系统/交易工具运营机构负责部署 CA 互认共享组件及其基础软件的资源服务器的安全运行管理。

（5）电子招标投标交易系统/交易工具运营机构应实时监测 CA 互认共享组件及其基础软件的运行情况，协调国家公共服务平台及时处理运行过程中的异常预警。电子招标投标交易系统/交易工具运营机构负责将 CA 互认共享组件的运行情况及时同步共享至国家公共服务平台和本省招标投标公共服务平台。

（6）电子招标投标交易系统/交易工具运营机构应及时监测本电子招标投标交易系统/交易工具中移动 CA 数字证书互认使用情况，并将移动 CA 数字证书互认应用的领域范围、开通的 CA 互认 APP、CA 机构、印章机构、供用户选择的付费模式、移动 CA 数字证书互认使用情况等数据及时同步共享至国家公共服务平台和本省招标投标公共服务平台。服务流程：

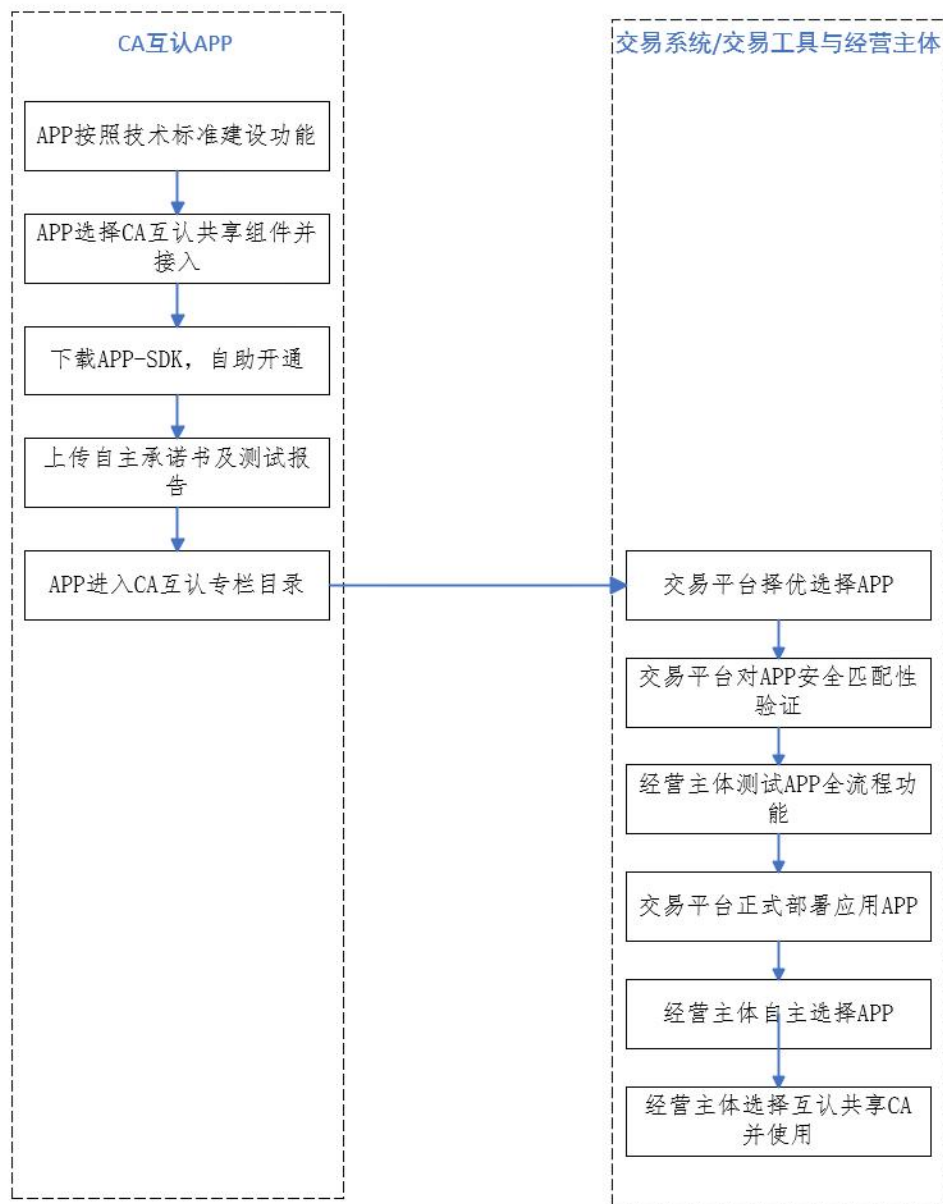


图 1 服务流程

主要流程：

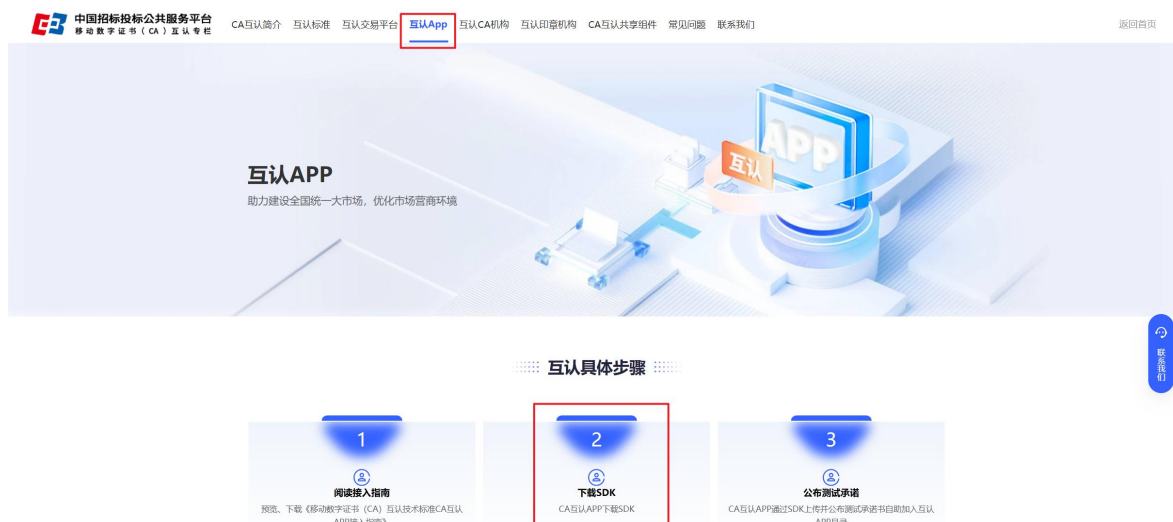
(1) CA 互认 APP 若提供移动 CA 数字证书互认服务，需按照《移动数字证书(CA)互认技术标准》开发建设，具备相应的功能，符合相关技术要求和数据内容要求；并按照《移动数字

证书（CA）互认技术标准》接入 CA 机构。

（2）CA 互认 APP 应当按照选择符合《移动数字证书（CA）互认技术标准》和开源代码的 CA 互认共享组件接入。

（3）在中国招标投标公共服务平台官网的 CA 互认专栏下载 SDK，通过 SDK 自助开通，如下图：





(4) CA 互认 APP 运营机构自主完成与 CA 互认共享组件的联调测试，联调测试通过 SDK 自主上传承诺书和测试报告后即进入 CA 互认共享目录。

(5) 电子招标投标交易系统/交易工具运营机构可以从 CA 互认共享目录中选择 CA 互认 APP，并开展相关安全匹配性验证，经营主体使用 APP 全功能验证通过后 CA 互认 APP 可在电子招标投标交易系统/交易工具上部署应用。

(6) 电子招标投标交易系统/交易工具用户可以自主选择 CA 互认 APP，并通过 CA 互认 APP 自主选择 CA 数字证书在电子招标投标交易系统/交易工具上共享使用。

2.CA 互认 APP 标准接口对接说明

2.1.对接准备

2.1.1.获取与 CA 互认共享组件对接的有关信息

2.1.1.1.获取 CA 互认 APP 唯一标识

CA 互认 APP 运营机构通过互认自助加入管理平台填写基本信息，在第二步填写 APP 加入信息页可找到 APP 编码，APP 编码也就是 APP 唯一标识。

2.1.1.2.获取电子招标投标交易系统/交易工具侧部署的 CA 互认共享组件地址

CA 互认 APP 运营机构在唯一标识获取成功后，可从国家公共服务平台获取 CA 互认共享目录内的在电子招标投标交易系统/交易工具一侧部署的 CA 互认共享组件列表及其唯一标识。

2.2.CA 互认 APP-SDK 对接说明

通过 APP-SDK 获取 APP 被交易平台开通的目录以及交易平台 CA 互认共享组件的 URL，URL 用于和交易平台 CA 互认共享组件进行标准接口进行数据交互。

2.2.1.对接前准备

2.2.1.1.环境要求

开发与运行环境保证以下条件：

(1) Java 运行环境:采用 JDK1.8 及以上版本,确保为电子交

易系统/交易工具的稳定运行提供坚实的基础。

(2) 网络连通性:与 CA 互认共享组件服务间需保持网络顺畅,确保数据交互无阻。

2.2.1.2.安装依赖

需要安装：cebpubservice-java-sdk-Core、cebpubservice-CAShare-java-sdk-ComponentsCatalog

Maven 依赖如下（version 以中央仓库最新版本为准）：

XML

<!--核心包依赖-->

<dependency>

<groupId>com.cebpubservice.cashare</groupId>

<artifactId>cebpubservice-java-sdk-Core</artifactId>

<version>1.0.0</version>

</dependency>

<!--app 侧 sdk-->

<dependency>

<groupId>com.cebpubservice.cashare</groupId>

<artifactId>cebpubservice-CAShare-java-sdk-ComponentsCatalog</artifactId>

<version>1.0.0</version>

```
</dependency>
```

2.2.2.调用说明

进行业务功能开发，需按照以下步骤实施：

2.2.2.1.SDK 初始化

需要在项目启动时初始化 SDK，为业务模块与 CA 互认共享组件交互做基础配置。需设置 `serviceUrl`（APP 侧 sdk 服务地址 `http(s):// 域名:端口/catalog`）、`clientId`（客户端 id）、`clientSecret`（客户端密钥）、`signatureSecret`（签名密钥），其中 `clientId`、`clientSecret`、`signatureSecret` 在互认自助加入管理平台填写基本信息后自动生成。

Java

```
import
com.cebpservice.cashare.sdk.componentscatalog.ComponentsCatalogSingleton;
import
org.springframework.context.annotation.Configuration;

import javax.annotation.PostConstruct;
```

```

@Configuration
public class Config {

    @PostConstruct
    public void init(){
        //项目启动时初始化互认共享组件地址
        /**参数说明
        *serviceUrl CA 互认共享组件地址
        *clientId 客户端 id
        *clientSecret 客户端密钥
        *signatureSecret 签名密钥
        */

        ComponentsCatalogSingleton.init("serviceUrl",
"clientId", "clientSecret", "signatureSecret");
    }

}

```

2.2.2.2.SDK 调用相关业务功能说明

```

Java

import
com.cebpubservice.cashare.sdk.componentscatalog.CatalogAndCo

```



```
componentsInfo;
import
com.cebpubservice.cashare.sdk.componentscatalog.param.BatchGe
tOpenCatalogAndComponentsInfoParamDTO;
import
com.cebpubservice.cashare.sdk.componentscatalog.param.GetOpen
CatalogAndComponentsInfoParamDTO;
import
com.cebpubservice.cashare.sdk.componentscatalog.result.GetOpen
CatalogAndComponentsInfoResultDTO;
import
com.cebpubservice.cashare.sdk.componentscatalog.result.IPageGet
OpenCatalogAndComponentsInfoResultDTO;
import com.cebpubservice.exceptions.ClientException;
import
org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;
import java.util.List;

@RestController
@RequestMapping("/demo")
```

```
public class CAShareDemoController {  
    /**  
     * 获取目录列表  
     * @return  
     * @throws ClientException  
     * @throws IllegalAccessException  
     */  
  
    @RequestMapping("/batchGetOpenCatalogAndComponentsInfoParamDTO")  
    public List<GetOpenCatalogAndComponentsInfoResultDTO>  
    getCAShareComponentsCatalog(){  
        try {  
            BatchGetOpenCatalogAndComponentsInfoParamDTO  
            batchGetOpenCatalogAndComponentsInfoParamDTO=new  
            BatchGetOpenCatalogAndComponentsInfoParamDTO();  
  
            batchGetOpenCatalogAndComponentsInfoParamDTO.setPageNu  
            m(1);  
  
            batchGetOpenCatalogAndComponentsInfoParamDTO.setPageSize
```

```

(10);

    //格式 yyyy-MM-dd HH:mm:ss

batchGetOpenCatalogAndComponentsInfoParamDTO.setUpdateTime("2024-01-01 23:59:59");

    IPageGetOpenCatalogAndComponentsInfoResultDTO
iPageGetOpenCatalogAndComponentsInfoResultDTO=CatalogAndComponentsInfo.batchGetOpenCatalogAndComponentsInfo(batchGetOpenCatalogAndComponentsInfoParamDTO);

    return

iPageGetOpenCatalogAndComponentsInfoResultDTO.getRecords(
);

} catch (ClientException e) {
    //处理异常,记录日志,并返回错误信息
    e.printStackTrace();
    return null;

}

}

/**
 * 获取目录详情

```

```

    * @return
    * @throws ClientException
    * @throws IllegalAccessException
    */

    @RequestMapping("/getOpenCatalogAndComponentsInfo")
    public          GetOpenCatalogAndComponentsInfoResultDTO
getCAShareComponentsCatalogInfo(){
    try {
        GetOpenCatalogAndComponentsInfoParamDTO
getOpenCatalogAndComponentsInfoParamDTO=new
GetOpenCatalogAndComponentsInfoParamDTO();

getOpenCatalogAndComponentsInfoParamDTO.setPlatformCode(
"A123456");

getOpenCatalogAndComponentsInfoParamDTO.setCaShareComp
onentsCode("B123455");

        return
CatalogAndComponentsInfo.getOpenCatalogAndComponentsInfo(
getOpenCatalogAndComponentsInfoParamDTO);
    } catch (ClientException e) {

```

```
//处理异常,记录日志,并返回错误信息
e.printStackTrace();
return null;
}
}
}
```

2.2.3.获取交易平台 CA 互认共享组件目录及详情

2.2.3.1.批量获取开通目录及详情

该接口用于批量获取交易平台开通的 CA 互认 APP 目录、CA 机构目录、印章机构目录及 CA 互认共享组件详情。

业务模块编码:

catalogAndComponentsInfo

功能接口编码:

batchGetOpenCatalogAndComponentsInfo

请求参数:

参数名称	类型	是否必填	描述
pageSize	int	是	每页数量
pageNum	int	是	页码
updateTime	date	否	更新时间(yyyy-MM-dd HH:mm:ss)

响应参数 data 列表：

参数名称	类型	描述
caShareComponents Code	String	ca 互认共享组件编码
caShareComponents Url	String	CA 共享组件回调 URL
platformPublicKey	String	交易平台公钥证书（通讯密 钥）
platformName	String	平台名称
platformCode	String	平台标识码
platformOrgName	String	运营机构名称
platformOrgCode	String	统一社会信用代码
platformOrgContact Name	String	联系人
platformOrgContact PhoneNumber	String	联系电话
version	String	数据版本号
allowNoUseSealOrg Mould		是否使用印章机构(1 允许不使 用 2 必须使用)
aesSecretKey	String	二维码生成密钥
platformRegion	String	省市区编码

参数名称	类型	描述
platformRegionName	String	省市区名称
systemList	List	交易系统列表
└ tradingSystemName	String	交易系统名称
└ tradingSystemCode	String	交易系统编码
└ tradingSystemUrl	String	交易系统 CA 互认入口地址
caOrgList	List	CA 机构列表
└ caOrgCode	String	CA 机构编码
└ caOrgName	String	CA 机构名称
└ caOperatorOrgName	String	运营机构名称
└ caOperatorOrgCode	String	统一社会信用代码
└ caOperatorOrgContactName	String	联系人
└ caOperatorOrgContact	String	联系电话

参数名称	类型	描述
ctPhoneNumber		
sealControlList	List	印章机构列表
└ sealControlSn	String	印章机构编码
└ sealControlOrgName	list	印章机构名称
└ sealControlLogo	String	印章机构图片
└ organizationName	String	运营机构名称
└ organizationCode	String	统一社会信用代码
└ contactName	String	联系人
└ contactPhone	String	联系电话

2.2.3.2.获取开通目录及详情

接口说明：

该接口用于单次获取 CA 互认 APP 被交易平台开通的目录及 CA 互认共享组件详情。

业务模块编码：

catalogAndComponentsInfo

功能接口编码：

getOpenCatalogAndComponentsInfo

参数名称	类型	是否必填	描述
platformCode	String	否	平台标识码
caShareComponentsCode	String	是	Ca 互认共享组件服务编码

响应参数：

参数名称	类型	描述
caShareComponents Code	String	ca 互认共享组件编码
caShareComponents Url	String	CA 共享组件回调 URL
platformPublicKey	String	交易平台公钥证书（通讯密 钥）
platformName	String	平台名称
platformCode	String	平台编码
platformOrgName	String	运营机构名称
platformOrgCode	String	统一社会信用代码
platformOrgContact Name	String	联系人
platformOrgContact PhoneNumber	String	联系电话
version	String	数据版本号
allowNoUseSealOrg Mould		是否使用印章机构(1 允许不使 用 2 必须使用)
aesSecretKey	String	二维码生成密钥
platformRegion	String	省市区编码

参数名称	类型	描述
platformRegionName	String	省市区名称
systemList	List	交易系统列表
└ tradingSystemName	String	交易系统名称
└ tradingSystemCode	String	交易系统编码
└ tradingSystemUrl	String	交易系统 CA 互认入口地址
caOrgList	List	CA 机构列表
└ caOrgCode	String	CA 机构编码
└ caOrgName	String	CA 机构名称
└ caOperatorOrgName	String	运营机构名称
└ caOperatorOrgCode	String	统一社会信用代码
└ caOperatorOrgContactName	String	联系人
└ caOperatorOrgContact	String	联系电话

参数名称	类型	描述
ctPhoneNumber		
sealControlList	List	印章机构列表
└ sealControlSn	String	印章机构编码
└ sealControlOrgName	String	印章机构名称
└ sealControlLogo	String	印章机构图片
└ organizationName	String	运营机构名称
└ organizationCode	String	统一社会信用代码
└ contactName	String	联系人
└ contactPhone	String	联系电话

2.2.3.3.设置接收开通目录及详情的服务地址

接口说明：

通过该接口设置接口开通目录及详情的回调地址，SDK 服务接收到更新数据，回调此接口。

业务模块编码：

catalogAndComponentsInfo

功能接口编码：

setReceiveUrl

请求参数：

参数名称	类型	是否必填	描述
receiveUrlList	List	是	
└receiveUrl	String	是	回调地址

响应参数：

无。

回调内容：

body 里 json 格式。

JSON

```
{
  "caComponentDTO": {
    "caShareComponentsCode": "ca 互认共享组件编码",
    "caShareComponentsUrl": "CA 共享组件回调 URL",
    "platformPublicKey": "交易平台公钥（通讯密钥）",
    "platformName": "平台名称",
    "platformCode": "平台编码",
    "platformOrgName": "运营机构名称",
    "platformOrgCode": "统一社会信用代码",
  }
}
```

```
"platformOrgContactName": "联系人",
"platformOrgContactPhoneNumber": "联系电话",
"version": "数据版本号",
"allowNoUseSealOrgMould": "是否使用印章机构(1 允许不
使用 2 必须使用)",
"aesSecretKey": "二维码生成密钥",
"platformRegion": "13,1301,130105",
"platformRegionName": "河北省,石家庄市,新华区"
},
"systemList": [
{
  "tradingSystemName": "测试系统",
  "tradingSystemCode": "A0301055469",
  "tradingSystemUrl": "www.abbs.com",
}
],
"caOrgList": [
{
  "caOrgCode": "CA 机构编码",
  "caOrgName": "CA 机构名称",
  "caOperatorOrgName": "运营机构名称",
```

```

        "caOperatorOrgCode": "统一社会信用代码",
        "caOperatorOrgContactName": "联系人",
        "caOperatorOrgContactPhoneNumber": "联系电话"
    }
],
"sealControlList": [
    {
        "sealControlSn": "印章机构编码",
        "sealControlOrgName": "印章机构名称",
        "sealControlLogo": "印章机构图片",
        "LorganizationName": "运营机构名称",
        "LorganizationCode": "统一社会信用代码",
        "LcontactName": "联系人",
        "LcontactPhone": "联系电话"
    }
]
}

```

2.3.CA 互认 APP 与 CA 互认共享组件对接

2.3.1.CA 互认 APP 互认标准接口说明

2.3.1.1.编码格式

统一编码格式为 UTF-8 编码。

2.3.1.2.协议规范

采用 HTTPS 协议作为接口通信的基础，确保数据传输的安全与高效。互认共享组件调用 CA 互认 App 应用服务端的统一地址为：“https://app 应用域名/CAShare/App”；

各 CA 互认 App 应用接入互认共享组件的统一地址为“https://互认共享组件域名/CAShare/Components”。

为了实现精准调用，每个功能模块的接口调用均需在 HTTP 请求头中设置特定的业务模块编码（ServiceCode）与功能接口编码（FeatureCode）。具体的编码值已在各接口说明文档中明确列出，便于开发者参照实施。

2.3.1.3.示例代码

以消息解密功能查询待解密用户信息接口这一功能为例，以下是代码片段，展示了如何在 HTTP 的 POST 请求中正确配置上述编码：

```
CloseableHttpClient httpClient = HttpClients.createDefault();
HttpPost httpPost = new HttpPost("待填写的互认共享组件统一地址");
// 设置请求头，包含业务模块与功能接口编码
httpPost.setHeader("ServiceCode", "messageDecrypt");
httpPost.setHeader("FeatureCode", "queryDecryptUserInfo");
//设置服务版本目前统一为 V1.0.0
```

```
httpPost.setHeader("Version","V1.0.0");
//请求唯一 id
httpPost.setHeader("Nonce", UuidUtil.getUuid());
//从 app 自主维护终端获取的互认 app 编码
httpPost.setHeader("AppCode", "互认 app 编码");
List<NameValuePair> pairs = new ArrayList<>();
Map param=new HashMap();
X509Certificate readCertificate = SM2Utils.readCertificate("分配的公钥证书");
String publicKey
=Base64.encode(readCertificate.getPublicKey().getEncoded());
param.put("tid","M11000000152a05ba6929494201bd191123b55b9e6a");
    //加密请求参数
String encrypt = SM2Utils.encrypt(publicKey,
JSON.toJSONString(param));
pairs.add(new BasicNameValuePair("businessData", encrypt));
UrlEncodedFormEntity entity = new UrlEncodedFormEntity(pairs,
"UTF-8");
httpPost.setEntity(entity);
HttpResponse response = httpClient.execute(httpPost);
```

```
String res = EntityUtils.toString(response.getEntity(), "UTF-8");
JSONObject jsonObject = JSON.parseObject(res);
String data = jsonObject.getString("data");
    //解密返回参数
String decrypt = SM2Utils.decrypt("分配的加解密私钥", data);
logger.info("解密后数据: " + decrypt)。
```

2.3.2.二维码解析及详情获取

扫码共享登录、扫码签名签章、以及扫码加密解密环节都需要获取二维码信息、对二维码解析、以及扫码状态同步和退出流程等操作,说明如下:

2.3.2.1.二维码内容解析

二维码的内容是经过国密 SM4 对称加密和 base64 编码后生成的数据。

内容格式如:

JSON

```
http://www.cebpublishservice.com?tradingSystemCode=111111&QR
CodeContent=a2V5JTNETDV1a1piRFBzR2lnd0lnZ29kNU5pUG
FjMWRDd0wrREdwK3FmZXhVUEZNYjhDRS9DZFRxMTRieG
diNkpTaVlwNzd0M3hwcmxlaWQwc01XbDNjbDZlVDE1THJkcU
Y2TjVOK21saTdmVldwdGJwano3cEhvNi9hTGhZY3BzRzh0UH
```

```
RPZzhGcUV1OWUwWUVVOXZRcHpzL3FXNzZmS1pjeXNpc  
W55NUE0RjNCKzRBJTNEJTNCYWxnb3JpdGhtJTNEc200JTN  
CdiUzRDE=
```

base64 解码 QRCodeContent 对应的内容格式如：

JSON

```
key=L5ukZbDPsGigwIggod5NiPac1dCwL+DGp+qfexUPFMb8C  
E/CdTq14bxgb6JSiYp77t3xprlej40sMWl3cl6ET15LrdqF6N5N+m  
li7fVWptbpjz7pHo6/aLhYcpsG8tPtOg8FqEu9e0YEU9vQpzs/qW  
76fKZcysiqny5A4F3B+4A=;algorithm=sm4;v=1
```

对 key 值进行 SM4（SM4/ECB/PKCS5Padding）解密后的格式为（对称密钥为 app-sdk 对接说明中"获取开通目录及详情"接口的 aesSecretKey 字段）：

JSON

```
qrCodeType=13;pid=A34013145205069578a42d548029d49cc75b  
4fe351a;tid=A34013145204a54c00ebcd04ba8adf44b868c505b6e;c  
aShareComponentsCode=10
```

其中，字段内容含义：

参数名称	类型	描述
qrCodeType	String	二维码类型（06 登录，10 撤章，14 授权解密，13 文件签章签名，16 授权加密，20 消息签名）
pid	String	连续签章标识(连续签章情况下使用)
tid	String	事件 id（平台标识码+UUID）
caShareComponentsCode	String	ca 互认共享组件编码

2.3.3.推送扫码状态

2.3.3.1.推送 CA 互认 APP 扫码状态

业务模块编码：

pushAppScannedStatus

功能接口编码：

pushQRCodeScannedStatus

CA 互认 APP 扫码后，CA 互认 APP 服务端向 CA 互认共享组件接口及时推送扫码状态，便于用户及时了解扫码进度。

请求参数：

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
pid	String	是	连续签章标识（签章时必须）
qrCodeType	String	是	二维码类型
scannedStatus	String	是	扫码状态：00 未扫，01 已扫（用户扫描二维码成功即为 01）

请求示例：

```

JSON
{
  "scannedStatus": "01",
  "qrCodeType": "06",
  "tid": "DCS2021060764c0bcdae7d548109a49af308e59b47b"
}

```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据

响应示例：

```

JSON
{
  "code": 200,
  "success": true,
  "data": null,
  "msg": "",
  "errorCode": null
}
```

2.3.3.2.推送 CA 互认 APP 退出状态

CA 互认 APP 扫码后，用户主动退出授权登录、签章、加解密等操作，需要将退出当前流程的事件通知到 CA 互认共享组件服务端，使用户在操作终端能及时收到反馈。

业务模块编码：

pushAppScannedStatus

功能接口编码:

pushExitStatus

请求参数:

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
pid	String	是	连续签章标识
msg	String	是	退出提示信息

请求示例:

```
JSON
{
  "msg": "APP 证书下载失败",
  "pid": "DCS2021060764c0bcdac7d548109a49af308e59b47b",
  "tid": "DCS2021060764c0bcdac7d548109a49af308e59b47b"
}
```

响应参数:

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据

响应示例：

```
JSON
{
  "code": 200,
  "success": true,
  "data": null,
  "msg": "",
  "errorCode": null
}
```

2.3.4.登录注册

二维码解析后，扫码的 CA 互认 APP 服务端向 CA 互认共享组件接口推送登录相关信息。

2.3.4.1.推送个人用户认证信息

个人用户登录时，扫码的 CA 互认 APP 服务端回调 CA 互

认共享组件接口推送个人登录信息。

业务模块编码：

sharedLoginRegistration

功能接口编码：

pushPersonalUserAuthInfo

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
personalName	String	是	用户姓名
idCardHash	String	是	身份证号 HASH
personalTransactionCode	String	是	个人唯一 ID（个人来自于每个 APP 的唯一标识，交易系统/交易工具可以根据此标识对登录的个人进行平台已有账户绑定,生成规则为 32 位 uuid）
telephoneNumber	String	是	手机号码
dataSignatureValue	String	是	对 tid 内容进行签名的数据签名值（rsa 算法:签名 hash 算法为 sha256，签名格式为 p7a，sm2 算法：签名 hash 算法为 sm3，签名格式为 p7a）
signatureData	String	是	待签名数据（取 tid 内容）

参数名称	类型	是否必填	描述
signatureCert	String	是	签名证书
publicKeyAlgorithm	String	是	公钥算法 01:RSA/00:SM2
idCardType	String	是	个人证件类型：01 身份证/99 其他
caOrgCode	String	是	CA 机构编码，详见： CA 机构编码
signatureCertSn	String	是	签名证书序列号

请求示例：

```
JSON
{
  "signatureData":
    "DCS20210607c3beb7643bc24844ad0cc14f1e372638",
  "dataSignatureValue":
    "MIID4AYKKoEcz1UGAQQCAqCCA9AwggPMAgEBMQ4...",
  "idCardType": "01",
  "telephoneNumber": "13552575156",
  "personalTransactionCode":
```

```
"CCF00918B11242558CC8222F219E6FFE",
  "publicKeyAlgorithm": "00",
  "idCardHash":
"eb7c8acdcc764b00bfe4d731f1759ba42998736b5146ae7616c6bf6f
e38e93f3",
  "signatureCertSn": "33000000010340226",
  "personalName": "朱一丹",
  "caOrgCode": "010001",
  "tid": "DCS20210607c3beb7643bc24844ad0cc14f1e372638",
  "signatureCert":
"MIICljCCAjqgAwIBAgIIMwAAABA0AiYwDAYIKoEcz1UB...
"
}
```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据

响应示例：

```

JSON
{
  "code": 200,
  "success": true,
  "data": null,
  "msg": "",
  "errorCode": null
}

```

2.3.4.2.推送单位用户认证信息

机构用户登录时，扫码的 CA 互认 APP 服务端回调 CA 互认共享组件接口推送当前机构用户信息。

业务模块编码：

sharedLoginRegistration

功能接口编码:

pushOrgUserAuthInfo

请求参数:

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
orgCode	String	是	单位代码
orgName	String	是	单位名称
personalName	String	是	用户姓名
idCardHash	String	是	用户身份证号 HASH
idCardType	String	是	个人证件类型：00 身份证 /01 其他
telephoneNumber	String	是	手机号码
dataSignatureValue	String	是	对 tid 内容进行签名的数据 签名值（rsa 算法:签名 hash 算法为 sha256，签名 格式为 p7a，sm2 算法：签 名 hash 算法为 sm3，签名 格式为 p7a）
signatureData	String	是	待签名数据（取 tid 内 容）
signatureCert	String	是	签名证书

参数名称	类型	是否必填	描述
publicKeyAlgorithm	String	是	公钥算法（RSA：01/SM2：00）
orgLicenceType	String	是	单位证照类型:97 统一社会信用代码/95 其他
caOrgCode	String	是	CA 机构编码，详见：CA 机构编码
signatureCertSn	String	是	签名证书序列号
orgTransactionCode	String	是	单位与用户关联的唯一 ID（交易系统/交易工具可以根据此标识对登录的单位进行平台已有账户绑定,生成规则为 32 位 uuid）

请求示例：

```

JSON
{
  "dataSignatureValue":
  "MIIEAQYKKoEcz1UGAQQCAqCCA/EwggPtA...",
  "orgName": "上海寻梦信息技术有限公司",
  "telephoneNumber": "13552575156",

```

```
"orgLicenceType": "01",
"publicKeyAlgorithm": "00",
"idCardHash":
"eb7c8acdcc764b00bfe4d731f1759ba42998736b5146ae7616c6bf6f
e38e93f3",
"signatureCertSn": "33000000010338506",
"tid": "DCS2021060776cd15c0feb5418ea4c7b2837ace5cff",
"orgTransactionCode":
"CCF00918B11242558CC8222F219E6FFE",
"signatureCert":
"MIICtzCCAlqgAwIBAgIIMwAAABAZhQYwDAYIKo...",
"signatureData":
"DCS2021060776cd15c0feb5418ea4c7b2837ace5cff",
"idCardType": "99",
"orgCode": "91310105090037252C",
"personalName": "朱一丹",
"caOrgCode": "010001"
}
```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据

响应示例：

```
JSON
{
  "code": 200,
  "success": true,
  "data": null,
  "msg": "",
  "errorCode": null
}
```

2.3.5.电子签章

文件电子签章支持个人签名和单位签章，个人签名指使用个人证书签署个人印章等；也支持单位签章，即使用单位证书签署单位有关印章等。

签章时，为简化用户体验，CA 互认 APP 需支持连续签章，即用户一次扫码可以多次落章，实现在不同位置盖多个章，直至用户在 CA 互认 APP 或签章工具退出。

2.3.5.1.推送单位用户签名证书

当单位用户电子签章时，CA 互认 APP 服务端向 CA 互认共享组件接口推送单位用户信息及签名证书。

业务模块编码：

electronicSealSignature

功能接口编码：

pushOrgUserSignatureCert

请求参数：

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
pid	String	是	连续签章标识(连续签章过程中，APP 需保证本标识不变)
orgName	String	是	单位名称
orgCode	String	是	单位代码
personalName	String	是	用户姓名
idCardHash	String	是	身份证号 HASH
publicKeyAlgorithm	String	是	公钥算法 01:RSA/00:SM2
signatureCert	String	是	签名证书
signatureCertSn	String	是	签名证书序列号
accessToken	String	是	授权令牌
orgTransactionCode	String	是	单位与用户关联的唯一 ID（生成规则为 32 位 uuid）
caOrgCode	String	是	CA 机构编码，详见： CA 机构编码

请求示例:

```
JSON
{
  "orgName": "上海寻梦信息技术有限公司",
  "orgCode": "91310105090037252C",
  "publicKeyAlgorithm": "00",
  "idCardHash":
"eb7c8acdcc764b00bfe4d731f1759ba42998736b5146ae7616c6bf6f
e38e93f3",
  "signatureCertSn": "33000000010338506",
  "pid":
"DCS20210607bbe14ce3b27a47e99be582870bcd3956",
  "personalName": "朱一丹",
  "caOrgCode": "010001",
  "accessToken":
"eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiI4ZWl1MGZhOW...",
  "tid": "DCS20210607bbe14ce3b27a47e99be582870bcd3956",
  "signatureCert":
"MIICtzCCAlqgAwIBAgIIMwAAABAAzhQYwDAYIKoEc...",
  "orgTransactionCode":
"C5BA5DE95CD149ECBE413724A2E5DB91"
```

```
}
```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据

响应示例：

```
JSON
{
  "code": 200,
  "success": true,
  "data": null,
  "msg": "",
  "errorCode": null
}
```

2.3.5.2.推送用户个人签名证书

当用户个人签名时，CA 互认 APP 服务端向 CA 互认共享

组件接口推送用户个人信息及签名证书。

业务模块编码：

electronicSealSignature

功能接口编码：

pushPersonalUserSignatureCert

请求参数：

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
pid	String	是	连续签章标识(连续签章过程中，APP 需保证本标识不变)
personalName	String	是	用户姓名
idCardHash	String	是	身份证号 hash
publicKeyAlgorithm	String	是	公钥算法 01:RSA/00:SM2
signatureCert	String	是	签名公钥证书
signatureCertSn	String	是	签名证书序列号
accessToken	String	是	授权令牌
personalTransactionCode	String	是	个人唯一 ID（个人来自于每个 APP 的唯一标识，生成规则为 32 位 uuid）
caOrgCode	String	是	CA 机构编码，详见： CA 机构编码

请求示例：

JSON

```
{
  "personalTransactionCode":
"CCF00918B11242558CC8222F219E6FFE",
  "publicKeyAlgorithm": "00",
  "idCardHash":
"eb7c8acdcc764b00bfe4d731f1759ba42998736b5146ae7616c6bf6f
e38e93f3",
  "signatureCertSn": "33000000010340226",
  "pid":
"DCS20210607e6ea925ba464412bb9aac4314d8c9536",
  "personalName": "朱一丹",
  "caOrgCode": "010001",
  "accessToken": "eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiI...",
  "tid": "DCS20210607e6ea925ba464412bb9aac4314d8c9536",
  "signatureCert":
"MIICljCCAjqgAwIBAgIIMwAAABA0Ai..."
}
```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据

响应示例：

```

JSON
{
  "code": 200,
  "success": true,
  "data": null,
  "msg": "",
  "errorCode": null
}
```

2.3.5.3.获取单位印章/个人印章列表

CA 互认共享组件根据签名签章主体信息，从 CA 互认 APP 服务端获取印章列表。

业务模块编码：

electronicSealSignature

功能接口编码：

getOrgUserOrPersonalUserSeals

请求参数：

请求格式 json。

参数名称	类型	是否必填	描述
tid	String	是	平台编码+UUID
pid	String	是	平台编码+UUID，签章 流程唯一标识
sealBelongType	String	是	证书所属类型 01 个人 /02 单位
orgTransactionCode	String	是	单位与用户关联的唯一 ID（生成规则为 32 位 uuid）
personalTransactionCode	String	是	个人唯一 ID（个人来自 于每个 APP 的唯一标 识）
idCardHash	String	是	身份证号 HASH
orgCode	String	是	单位代码
tradingSystemCode	String	是	交易系统编码
accessToken	String	是	授权令牌

请求示例：

JSON

```
{  
  "personalTransactionCode":  
"CCF00918B11242558CC8222F219E6FFE",  
  "tradingSystemCode": "DCS20210607",  
  "pid":  
"DCS20210607e6ea925ba464412bb9aac4314d8c9536",  
  "accessToken": "eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiI...",  
  "idCardHash":  
"eb7c8acdcc764b00bfe4d731f1759ba42998736b5146ae7616c6bf6f  
e38e93f3",  
  "orgCode": "91310105090037252C",  
  "sealBelongType": "01",  
  "tid": "DCS20210607e6ea925ba464412bb9aac4314d8c9536"  
}
```

响应参数:

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据

响应示例：

```

JSON
{
  "code": 200,
  "success": true,
  "data":
"04A146372539A91C3CBA3057A72B4FD0412673D7B4C8A22F
FE8D51C01389DAFD3F5C251448B64799B5ABFC6FFD9F7211
10E8AAF8895C4E439F0D216172BFA81C02F0E9BC7A554A3D
1FE2E1CF4E287925C480CC9B8E0FB56D8DE2E56AB093B85E
FCFF64",
  "msg": "",
  "errorCode": null
}

```

data 解密后的数据：

参数名称	类型	描述
	array	
L sealType	String	印章类型 00 手写签名/01 公章 /02 合同章/03 财务章/04 人名章 /05 法定代表人章
L sealImage	String	印章图片
L sealBelongType	String	印章所属类型 01 个人/02 企业
L sealUrl	String	印章预览地址
L sealName	String	印章名称
L sealSn	String	印章编号

2.3.5.4.获取单位印章/个人印章信息

CA 互认共享组件根据签名签章主体信息，从 CA 互认 APP 服务端获取印章图片符合 GBT/38540-2020 规范的印章结构体。

业务模块编码：

electronicSealSignature

功能接口编码：

getOrgUserOrPersonalUserSealInfo

请求参数：

参数名称	类型	是否必填	描述
sealSn	String	是	印章编号
signatureCertSn	String	是	CA 证书序列号
accessToken	String	是	授权令牌
tid	String	是	事件 ID（平台标识码+UUID）
caOrgCode	String	是	CA 机构编码，详见： CA 机构编码

请求示例：

```

JSON
{
  "sealSn": "1f6cde69165443ed811c7de6899fd506",
  "signatureCertSn": "3300000010340226",
  "accessToken":
  "eyJhbGciOiJIUzI1NiJ9.eyJqdGkiOiJjZjQas....",
  "tid": "DCS20210607e6ea925ba464412bb9aac4314d8c9536",
  "caOrgCode" : "010001"
}

```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

data 解密后的数据：

参数名称	类型	描述
sealInfo	String	电子印章信息的 Base64 编码

响应示例：

```

JSON
{
  "code": 200,
  "success": true,
  "data":
  "04A146372539A91C3CBA3057A72B4FD0412673D7B4C8A22F
  FE8D51C01389DAFD3F5C251448B64799B5ABFC6FFD9F7211
  10E8AAF8895C4E439F0D216172BFA81C02F0E9BC7A554A3D
  1FE2E1CF4E287925C480CC9B8E0FB56D8DE2E56AB093B85E
  FCFF64",

```

```
"msg": "",  
"errorCode": null  
}
```

2.3.5.5.上传待签名原文

CA 互认共享组件向 CA 互认 APP 服务端推送文件哈希。

业务模块编码：

electronicSealSignature

功能接口编码：

uploadSignatureOriginalText

请求参数：

参数名称	类型	是否必填	描述
pid	String	是	连续签章标识
tid	String	是	事件 ID（平台标识码+UUID）
sealSn	String	是	印章编号
electronicSignature UserType	String	是	签章用户类型 （01 个人/02 单位）
orgTransactionCode	String	是	单位与用户关联的唯一 ID（生成规则为 32 位 uuid）
personalTransaction Code	String	是	个人唯一 ID
tradingSystemCode	String	是	交易系统编码
accessToken	String	是	授权令牌
signContinued	String	是	是否连续签章 00:单次签章、 01：连续签章
signatureOriginalTextList	array	是	待签名原文列表

参数名称	类型	是否必填	描述
LsignatureOriginal TextId	String	是	待签名原文唯一 ID
LsignatureOriginal Text	String	是	待签名原文
sealType	String	是	印章类型 00 手 写签名/01 公章 /02 合同章/03 财 务章/04 人名章 /05 法定代表人 章
sealSn	String	是	印章编号

请求示例：

```

JSON
{
  "pid":
  "DCS20210607e6ea925ba464412bb9aac4314d8c9536",
  "tid": "DCS20210607e6ea925ba464412bb9aac4314d8c9536",
  "signatureOriginalTextList": [{
    "signatureOriginalTextId":
    "1d9eae56b03a4cd3b3adbdde62fb08ca",

```

```
    "signatureOriginalText":
    "MIIFBAIBBDCCBMcwggLeMA8WakVTA gE..."
  }],
  "sealSn": "1f6cde69165443ed811c7de6899fd506",
  "sealType": "00",
  "electronicSignatureUserType": "01",
  "personalTransactionCode":
  "CCF00918B11242558CC8222F219E6FFE",
  "tradingSystemCode": "DCS20210607",
  "accessToken": "eyJhbGciOiJIUzI1NiJ9...",
  "signContinued": "00"
}
```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

data 解密后的数据：

参数名称	类型	描述
└ tid	String	事件 ID（平台标识码+UUID）

响应示例：

```

JSON
{
  "code": 200,
  "success": true,
  "data":
    "04A146372539A91C3CBA3057A72B4FD0412673D7B4C8A22F
    FE8D51C01389DAFD3F5C251448B64799B5ABFC6FFD9F7211
    10E8AAF8895C4E439F0D216172BFA81C02F0E9BC7A554A3D
    1FE2E1CF4E287925C480CC9B8E0FB56D8DE2E56AB093B85E
    FCFF64",
  "msg": "",
  "errorCode": null
}

```

2.3.6.推送待签名原文签名值

CA 互认 APP 的用户使用私钥签名后，CA 互认 APP 服务端向 CA 互认共享组件接口推送多签名值信息。

业务模块编码：

electronicSealSignature

功能接口编码:

pushOriginalTextSignatureValue

请求参数:

参数名称	类型	是否必填	描述
tid	String	是	事件 ID
signatureOriginalTextList	array	是	待签名原文列表
└ signatureOriginalTextId	String	是	待签名原文唯一 ID
└ signatureOriginalText	String	是	待签名原文
└ originalTextSignatureValue	String	是	待签名原文签名值, rsa 算法:签名 hash 算法为 sha256, 签名格式为 p1, sm2 算法: 签名 hash 算法为 sm3, 签名格式为 p1

请求示例:

JSON

```
{
  "signatureOriginalTextList": [{
    "originalTextSignatureValue":
    "MEUCIQCC8faDpmrhzmzSSv...",
    "signatureOriginalText":
    "MIIFBAIBBDCCBMcwggLeMA8Wak...",
    "signatureOriginalTextId":
    "1d9eae56b03a4cd3b3adbdde62fb08ca"
  }],
  "tid": "DCS2021060751c8d278339346f0953c9550131fa62a"
}
```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

响应示例：

JSON

```
{  
  "code": 200,  
  "success": true,  
  "data": null,  
  "msg": "",  
  "errorCode": null  
}
```

2.3.6.1.退出签章

当用户在退出签章时 CA 互认共享组件向 CA 互认 APP 服务端接口推送退出指令。

业务模块编码：

electronicSealSignature

功能接口编码：

exitElectronicSealSignature

请求参数：

参数名称	类型	是否必填	描述
pid	String	是	连续签章标识(连续签章过程中，APP 需保证本标识不变)
tid	String	是	值与 PId 保持一致
accessToken	String	是	授权令牌
source	String	是	指令来源 01：用户发起，02：系统发起

请求示例：

```

JSON
{
  "pid": "DCS202106077a47f77036a64c66a2865aa2b72fc86b",
  "source": "01",
  "accessToken": "eyJhbGciOiJIUzI1NiJ9.eyJ...",
  "tid": "DCS202106077a47f77036a64c66a2865aa2b72fc86b"
}

```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

响应实例：

```

JSON
{
  "code": 200,
  "success": true,
  "data": null,
  "msg": "",
  "errorCode": null
}
```

2.3.7.撤销电子签章

2.3.7.1.查询签名 证书序列号

撤章过程时，需要使用当前签章的证书做签名来确认用户操作，本接口用于 **ca 互认互认 app** 从 **ca 互认共享组件** 获取需要撤销签章的证书序列号。

业务模块编码：

revokeElectronicSealSignature

功能接口编码:

querySignatureCertSn

请求参数:

参数名称	类型	是否必填	描述
tid	String	是	事件 ID (平台标识码+UUID)

请求示例:

```
JSON
{
  "tid":
  "A340131452001f4165f520b451d8af34e222201179a"
}
```

响应参数:

参数名称	类型	描述
code	Integer	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	String	加密数据字符串

data 解密后的数据:

参数名称	类型	描述
tid	String	平台标识码+UUID
signatureCertSn	String	签章序列号

响应示例：

JSON

```
{
  "code": 200,
  "success": true,
  "data":
    "04A146372539A91C3CBA3057A72B4FD0412673D7B4C8
    A22FFE8D51C01389DAFD3F5C251448B64799B5ABFC6F
    FD9F721110E8AAF8895C4E439F0D216172BFA81C02F0E
    9BC7A554A3D1FE2E1CF4E287925C480CC9B8E0FB56D8
    DE2E56AB093B85EFCFF64",
  "msg": "",
  "errorCode": null
}
```

2.3.7.2.推送数据签名值

CA 互认 APP 的用户授权撤章后，CA 互认 APP 服务端向 CA 互认共享组件接口推送撤章信息。

业务模块编码：

revokeElectronicSealSignature

功能接口编码：

pushDataSignatureValue

请求参数：

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
signatureCert	String	是	签名证书
dataSignatureValue	String	是	对 tid 内容进行签名的数据签名值（rsa 算法:签名 hash 算法为 sha256，签名格式为 p7a，sm2 算法：签名 hash 算法为 sm3，签名格式为 p7a）
signatureData	String	是	待签名数据（取 tid 内容）
publicKeyAlgorithm	String	是	公钥算法 01:RSA/00:SM2

请求示例：

JSON

```

{
  "signatureData":
"DCS20210607e24c15f8b79141258187203fc9cd53c1",
  "dataSignatureValue":
"yk+C9+2Nf+sUOzYq7XKNgU1BY5...",
  "publicKeyAlgorithm": "00",
  "tid": "DCS20210607e24c15f8b79141258187203fc9cd53c1",
  "signatureCert":
"MIICljCCAjqgAwIBAgIIMwAAABA0Ai..."
}

```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

响应示例：

```

JSON
{

```

```
"code": 200,  
"success": true,  
"data": null,  
"msg": "",  
"errorCode": null  
}
```

2.3.8.消息签名

2.3.8.1.查询签名消息

业务模块编码:

messageSignature

功能接口编码:

queryMessageSignature

请求参数:

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）

请求示例:

JSON

```
{  
  "tid": "A340131452001f4165f520b451d8af34e222201179a"
```



```
}
```

响应参数:

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

data 解密后的参数:

参数名称	类型	描述
tid	String	平台标识码+UUID
signatureMessage	String	需要签名的消息原文

```
{
  "code": 200,
  "success": true,
  "data":
    "04A146372539A91C3CBA3057A72B4FD0412673D7B4C8A22F
    FE8D51C01389DAFD3F5C251448B64799B5ABFC6FFD9F7211"
```

```
10E8AAF8895C4E439F0D216172BFA81C02F0E9BC7A554A3D
1FE2E1CF4E287925C480CC9B8E0FB56D8DE2E56AB093B85E
FCFF64",
    "msg": ""
}
```

2.3.8.2.推送消息签名值

CA 互认 APP 的用户使用私钥签名后，CA 互认 APP 服务端向 CA 互认共享组件接口推送签名值。

业务模块编码：

messageSignature

功能接口编码：

pushMessageSignatureValue

请求参数：

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
signatureCertSn	String	是	证书序列号
signatureCert	String	是	签名证书
messageSignatureValue	String	是	对 tid 内容进行签名的数据签名值（rsa 算法:签名 hash 算法为 sha256，签名格式为 p7a，sm2 算法：签名 hash 算法为 sm3，签名格式为 p7a）
signatureMessage	String	是	待签名数据（上一个接口中获取的需要签名的消息原文）
publicKeyAlgorithm	String	是	公钥算法 01:RSA/00:SM2
caOrgCode	String	是	CA 机构编码，中国招标投标公共服务平台统一赋码
signatureUserType	String	是	签名用户类型（01 个人/02 单位）

请求示例:

```
JSON

{
  "tid": "M1100015d2226dc39af24d82934bb61de61a728b",
  "signatureCertSn":
    "6FCB2D2833C60FFD86BD715FEE83ABE8381E313A",
  "signatureCert": "MIIC9DCCApmgAwIBAgIUXq...",
  "messageSignatureValue": "MIIF8AYJKoZIh....",
  "signatureMessage": "123",
  "caOrgCode": "010001",
  "publicKeyAlgorithm": "01",
  "signatureUserType": "01"
}
```

响应参数:

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

响应示例：

```
{
  "code": 200,
  "success": true,
  "data": null,
  "msg": ""
}
```

2.3.9.消息加密

适用于文件信封加密和消息加密。

2.3.9.1.推送单位用户加密证书

当 CA 互认 APP 用户使用单位证书加密时，CA 互认 APP 服务端向 CA 互认共享组件接口推送单位的加密证书公钥。

业务模块编码：

messageEncrypt

功能接口编码：

pushOrgUserEncryptionCert

请求参数:

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
orgCode	String	是	单位代码
orgName	String	是	单位名称
idCardHash	String	是	证件号码哈希值 (即身份证号 Hash 值,使用国密 SM3 摘要算法加密)
personalName	String	是	用户姓名
telephoneNumber	String	是	手机号码（加密人）
caOrgCode	String	是	CA 机构的编号
encryptionCertSn	String	是	加密证书序列号
encryptionCert	String	是	加密证书
signatureCert	String	是	签名证书
dataSignatureValue	String	是	对 tid 内容进行签名的数据签名值 (rsa 算法:签名 hash 算法为

参数名称	类型	是否必填	描述
			sha256，签名格式为 p7a，sm2 算法：签名 hash 算法为 sm3，签名格式为 p7a)
signatureData	String	是	待签名数据（取 tid 内容）
signatureCertSn	String	是	签名证书序列号
publicKeyAlgorithm	String	是	公钥算法（RSA：01/SM2：00）

请求示例：

```

JSON
{
  "dataSignatureValue":
  "MIIHUwYJKoZIhvcNAQcCoIIHRD...",
  "telephoneNumber": "13261560102",
  "orgName": "gless department",
  "publicKeyAlgorithm": "01",
  "idCardHash":
  "8b0da759ff9fdea6203844fc4202566a5c39b1d94ac531c0a33111d

```



```
77509c26e",
  "signatureCertSn":
"13735FB995F1E6EC7C99C83F10415D67D0A46F5B",
  "encryptionCert": "MIIFPDCCBCSgAwIBAgIUWxyj9...",
  "tid": "DCS202106077183729ad232444ab6233d5432ac09aa",
  "signatureCert": "MIIFJTCCBA2gAwIBAgIUUE3NfuZX...",
  "signatureData":
"DCS202106077183729ad232444ab6233d5432ac09aa",
  "orgCode": "913301007882728281",
  "personalName": "张小兽",
  "caOrgCode": "010002",
  "encryptionCertSn":
"5B1CA3F5B624CA8CCCEA03CC28213779366B9AC2",
  "encryptionUserType": "02"
}
```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

响应示例：

JSON

```
{
  "code": 200,
  "success": true,
  "data": null,
  "msg": ""
}
```

2.3.9.2.推送个人用户加密证书

当 CA 互认 APP 用户使用个人证书加密时，CA 互认 APP 服务端向 CA 互认共享组件接口推送用户个人加密证书公钥。

业务模块编码：

messageEncrypt

功能接口编码：

pushPersonalUserEncryptionCert

请求参数:

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
idCardHash	String	是	身份证号 HASH
personalName	String	是	加密人姓名
telephoneNumber	String	是	加密人手机号码
caOrgCode	String	是	CA 机构的编号
encryptionCertSn	String	是	加密证书序列号
encryptionCert	String	是	加密证书
signatureCert	String	是	签名证书
dataSignatureValue	String	是	对 tid 内容进行签名的数据签名值（rsa 算法:签名 hash 算法为 sha256，签名格式为 p7a，sm2 算法:签名 hash 算法为 sm3，签名格式为 p7a）
signatureData	String	是	待签名数据（取 tid 内容）

参数名称	类型	是否必填	描述
signatureCert Sn	String	是	签名证书序列号
publicKeyAlg orithm	String	是	公钥算法（RSA： 01/SM2：00）

请求示例：

```

JSON
{
  "dataSignatureValue":
  "MIIHSAYJKoZIhvcNAQcCoIIHOTCCBzU...",
  "telephoneNumber": "13261560102",
  "orgName": "null",
  "publicKeyAlgorithm": "01",
  "idCardHash":
  "b8f80b8f0e4590becf171510210b1b08bac040440be28c069206225
  543f69c7b",
  "signatureCertSn":
  "6FCB2D2833C60FFD86BD715FEE83ABE8381E313A",
  "encryptionCert": "MIIFMTCCBBmgAw...",
  "tid": "DCS20210607b1e3b664e7314f38b2005802a4a3141b",
  "signatureCert": "MIIFGjCCBAKgAwIBAgIUb8...",

```

```
"signatureData":
"DCS20210607b1e3b664e7314f38b2005802a4a3141b",
  "orgCode": "null",
  "personalName": "张小兽",
  "caOrgCode": "010002",
  "encryptionCertSn":
"2ED98476E884AD5E3E1109AF57A3F73D0DD83173",
  "encryptionUserType": "01"
}
```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

响应示例：

```
{
  "code": 200,
  "success": true,
  "data": null,
```

```
"msg": ""  
}
```

2.3.10.消息解密

适用于文件解密和消息解密。

2.3.10.1.查询待解密用户信息

CA 互认 APP 在做解密时，互认 APP 可以调用本接口获取当前解密操作对应的解密用户或者统一社会信用代码，以校验当前待解密内容是否与当前操作用户是否匹配。

业务模块编码：

messageDecrypt

功能接口编码：

queryDecryptUserInfo

请求参数：

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）

请求示例：

```
JSON  
  
{  
  "tid": "A34013145200970b85e90ca4da9a4cdeba28fe76384"
```

```
}
```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

data 解密后的数据：

参数名称	类型	是否必填	描述
idCardHash	String	是	证件号码哈希
decryptionUserType	String	是	解密用户类型（01:个人 02:单位）
orgCode	String	是	统一社会信用代码

响应示例：

```
JSON
{
  "code": 200,
  "success": true,
  "data":
    "04A146372539A91C3CBA3057A72B4FD0412673D7B4C8A22F"
```



```
FE8D51C01389DAFD3F5C251448B64799B5ABFC6FFD9F7211
10E8AAF8895C4E439F0D216172BFA81C02F0E9BC7A554A3D
1FE2E1CF4E287925C480CC9B8E0FB56D8DE2E56AB093B85E
FCFF64",
  "msg": ""
}
```

2.3.10.2.推送解密用户信息

推送用户信息到 CA 互认共享组件。

业务模块编码：

messageDecrypt

功能接口编码：

pushDecryptUserInfo

请求参数：

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
accessToken	String	是	授权令牌
decryptionUserType	String	是	解密用户类型（01:个人 02:机构）
idCardHash	String	是	证件号码哈希值(即身份证号 Hash 值,使用国密 SM3 摘要算法加密,加密时生成在加密信封 xml 中)
orgCode	String	是	单位代码

请求示例：

```

JSON
{
  "tid": "A34013145200970b85e90ca4da9a4cdeba28fe76384",
  "accessToken": "7d0b6a6ff39e4f0c899816fbd2c40020",
  "idCardHash":
"8b0da759ff9fdea6203844fc4202566a5c39b1d94ac531c0a33111d
77509c26e",
  "decryptionUserType" : "01"

```

```
"orgCode": "92370104MA3LG9TX81"
}
```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

响应示例：

JSON

```
{
  "code": 200,
  "success": true,
  "data": null,
  "msg": ""
}
```

2.3.10.3.上传加密密钥信封

CA 互认共享组件设置待解密信息，推送至 CA 互认 APP

端。文件解密时，待解密信息为信封，消息解密时，待解密信息为消息密文。

业务模块编码：

messageDecrypt

功能接口编码：

uploadEncryptedKeyDataEnvelope

请求参数：

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
accessToken	String	是	授权令牌
encryptedKeyDataEnvelopeList	array	是	信封列表
L encryptedKeyDataEnvelopeId	String	是	加密密钥信封唯一标识（业务平台可自定义生成或使用加密密钥信封文件中的对应参数）
L caOrgCode	String	是	CA 机构编码，详见：CA 机构编码
L encryptionCertSn	String	是	加密证书序列号
L encryptionKey	String	是	加密后的密钥（对称密钥的加密密文，对称密钥用于文件加密）
L publicKeyAlgorithm	String	是	公钥算法（00:SM2 01:RSA）

请求示例:

JSON

```
{
  "tid": "A34013145200970b85e90ca4da9a4cdeba28fe76384",
  "accessToken": "7d0b6a6ff39e4f0c899816fbd2c40020",
  "encryptedKeyDataEnvelopeList": [{
    "encryptedKeyDataEnvelopeId": "BC1A11627E004",
    "caOrgCode": "010001",
    "encryptStr":
      "MIIEOzCCAyOgAwIBAgIIMwAAAAInd5MwDQYJKoZIhvcN
      AQELBQAwxTELMaKGA1UEBhMCQ04xMDAuBgNVBAoMJ
      0NoaW5hIEZpbmFuY2lhbCBDZXJ0aWZpY2F0aW9uIEF1dGhvc
      ml0eTEcMBoGA1UEAwwTQ0ZDQSBBQ1MgVEVTVCBPQ0Ez
      MzAeFw0yMjEyMDEwNDJhFw0yMzEwMjUwMjIwMTha
      MIGaMQswCQYDVQQGEwJDTjEXMBUGA1UECgwOQ0ZDQ
      SBPQ0EzMyBSU0ExDTALBgNVBAsMBFNERUcxGTAXBgN
      VBAsMEE9yZ2FuaXphdGlvbmFsLTlxSDBGBgNVBAMMP1NE
      RUdA6ams6ZKi6K+a5YW06YeR5bGe6LWE5rqQ5pyJ6ZmQ5Y
      Ws5Y+4QE45MTM0MDUwME1BMlQwN1IDMFhAMjCCASIw
      DQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBANwNy0n
      ypFX7CoOxVz1BFRbw9p87midcGeny8VdTQ7cFMjyVc1aVVp8
```

```
PQo/0FJoRhaEXV17/tTZhcdleqRHQc0thJwZIfSyFnuosBgyPjJIP
AedmNg0DOsc7NOhGawyWrMEBaKeL1mwNxoxEACn5zgbTf
Vln6zUMP/6jrBEa1iWD4fumZRZ+KnCkNCFbTQm6vlibYrH0W
kxmHAwiEzT7+nJUmlYk2JWtZX9OD52w6MpxibKH84YJ42tMf
7VvTdnScXooBE4Rk8a5KAUWAj1+rZVqrd8vco8bSA03XmrLZ
Xgw9ZQIVZZ9XYr1/btlSiRBheCuNe7mOfZy7TJrqiF/0q/vIWNX
5UnbRXz3omMRii3fR/Zpjek17v+WhRNl0tQ13JKtFb6q5UGrAN
iYCCAoMpYq+tYLKE/9r8Aijn+nrzuR2ABFaSsbh9DOrv26EKQs
vIHnYTTvlnWquSP/6B1ij8hJe5CDLK4FUor2mHEBVEa5GqlZN
BfgLVUqoSVlojWs2hYgpqaGP0aEjI+7GUKvABhrSWAoKvQTS
1rC6knOd/WJnfSgU9Akig+/PLGryXvw==",
"encryptionCertSn": "33000000977793",
"publicKeyAlgorithm": "01",
"idCardHash": "566420cee02d0944f09e3b284ed208c"
}]
}
```

响应参数：

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

响应示例：

JSON

```
{
  "code": 200,
  "success": true,
  "data": null,
  "msg": ""
}
```

2.3.10.4.推送解密后的数据

用户通过 CA 互认 APP 解密成功后，CA 互认 APP 服务端向 CA 互认共享组件接口推送已解密信息。

业务模块编码：

messageEncrypt

功能接口编码：

pushDecryptedData

请求参数：

参数名称	类型	是否必填	描述
tid	String	是	事件 ID（平台标识码+UUID）
decryptInfoList	array	是	解密信息集合
LdecryptKey	String	是	解密后的明文密钥(明文对称密钥用于文件解密)
LencryptedKeyDataEnvelopeId	String	是	加密密钥信封的唯一标识（业务平台可自定义生成或使用加密密钥信封文件中的对应参数）
LdecryptStatus	String	是	解密状态（该字段表示解密成功或解密失败，0 表示解密成功，-1 表示解密失败，-2 表示 APP 未找到证书，-3 表示该标段非当前 App 账号加密,请使用原加密 App 账号进行解密，-4 表示用户 APP 端取消解密）

请求示例：

JSON

```
{
  "tid": "A34013145200970b85e90ca4da9a4cdeba28fe76384",
  "decryptInfoList": [{
    "encryptedKeyDataEnvelopeId": "AH-0001",
    "decryptedKey": "b1zxas1OQ2CZ",
    "decryptStatus": "0"
  }]
}
```

响应参数:

参数名称	类型	描述
code	String	响应状态(200/400)
success	boolean	true 成功;false 失败
msg	String	状态结果说明
data	string	加密数据字符串

响应示例:

JSON

```
{
  "code": 200,
  "success": true,
```

```
"data": null,  
"msg": ""  
}
```

3.自主联调测试进入 CA 互认共享目录

CA 互认 APP 应按照《移动数字证书(CA)互认技术标准》要求进行研发，并与 CA 数字证书组合运营。CA 互认 APP 与 CA 互认共享组件自主联调测试，通过 SDK 上传承诺书和测试报告后即可进入 CA 互认专栏，供电子招标投标交易系统/交易工具的运营机构选择使用。

4.附录

4.1.附录 1:接口间调用数据加密解密约定

(1) CA 互认 APP 与 CA 互认共享组件交互时需要将数据明文进行非对称加密。具体流程为：根据上述接口中参数将数据转为 json 字符串，使用 APP 注册时提供的公钥进行 SM2 加密封装为 businessData 字段，并通过 post 请求以 formdata 方式发送到共享 CA 互认共享组件。

(2) CA 互认共享组件调用 APP 服务端接口时采用相同的加密方式。

(3) 请求头中需要设置 Nonce 字段（代表本次请求的唯一 id，可以是 uuid），appCode 字段(CA 互认 APP 唯一标识,CA 互认 APP 调用 CA 互认共享组件服务时需要上送)。

(4) 接口通讯中业务参数加解密方式中 SM2Utils 部分示例代码。

Java

```
import cn.hutool.core.codec.Base64;
import cn.hutool.crypto.asymmetric.KeyType;
import cn.hutool.crypto.asymmetric.SM2;
import org.bouncycastle.jce.provider.BouncyCastleProvider;

import java.io.ByteArrayInputStream;
import java.io.IOException;
import java.security.NoSuchProviderException;
import java.security.Security;
import java.security.cert.CertificateException;
import java.security.cert.CertificateFactory;
import java.security.cert.X509Certificate;

public class SM2Utils{
    static {
        // 添加 BouncyCastle 提供者，支持 SM2 算法
        Security.addProvider(new BouncyCastleProvider());
    }
}
```

```

    }

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

        String str =
            "{\"Tid\":\"M11000000152a05ba6929494201bd191123b55b9e6a\"
            ,\"Pid\":\"M11000000159248fd1ab97c4baaa45ed82eff953fdd\", \"QRCodeType\":\"01\", \"isScanned\":\"01\", \"appSource\":\"00000\"}";

        System.out.println("参数:" + str);
        //

        //私钥
        String
privateKey="MIGTAgEAMBMGBYqGSM49AgEGCCqBHM9VA
YItBHkwdwIBAQQg+Q99Rq2XJTE4pBBOaXg3PaTvWStPxSP7
ARmRjLq09YagCgYIKoEcz1UBgi2hRANCAARBQ9FOCyEhpd
UWPaeO5YlgKC9e9v/um4bdK0CHxHL77mIMfJvIFvaYK0QJU
DdBbwlzqJTj7osI0naNduiTvqfv";

        //公钥证书
        String
certBase64="MIIBrjCCAVWgAwIBAgIGA ZEMnmfaMAoGCCq
BHM9VAYN1MHwx CzAJBgNVBAYTAkNOMU8wTQYDVQQ

```

DDEZCSkdYQOS4reWbveaLm+agh+WFrOWFseacjeWKoeW5s
+WPsOaciemZkOWFrOWPuEBOTjKxMTEwMTA4MDg5NjM4
MTIYUUAxMRwwGgYDVQQpDBNOOTExMTAxMDgwODk2
MzgxOVhRMB4XDTI0MDgwMTA2MjYzM1oXDTI1MDgwMT
A2MjYzM1owQTEbMBkGA1UEBRMSOTExMTAxMDgwODk
2MzgxOVhRMRUwEwYDVQQKDAzmtYvor5Xlhazlj7gxCzAJB
gNVBAYTAkNOMFkwEwYHKoZIzj0CAQYIKoEcz1UBgi0DQ
gAEQUPTgshIaXVFj2njuWJYCgvXvb/7puG3StAh8Ry++5iDH
yb5Rb2mCtECVA3QW8Jc6iU4+6LCNJ2jXbok76n7zAKBggqgR
zPVQGDdQNHADBEAiBwlY2zQtN5KL1P82Ru4DZ867Cxv8R
EY9t7FKdIQR7WpgIgJFldIEYuYaTL3N18l9WidmTRgl2+JCX0
S3b9MQf8BJw=";

//解析子证书，获取公钥

```
X509Certificate readCertificate =  
readCertificate(certBase64);  
  
System.out.println(readCertificate.toString());  
  
String publicKey =  
Base64.encode(readCertificate.getPublicKey().getEncoded());  
  
System.out.println("读取公钥:" + publicKey);  
  
X509Certificate readRootCertificatesub =  
readCertificate(certBase64);
```

```
//加密测试  
String certEncrypt = encrypt(publicKey, str);  
System.out.println("加密密文:" + certEncrypt);  
System.out.println("解密原文:" + decrypt(privateKey,  
certEncrypt));
```

```
String caCertBase64 =  
"MIIEuDCCBF2gAwIBAgIKGhAAAAAABNbBMzAKBggqgRz  
PVQGDdTBEMQswCQYDVQQGEwJDTjENMA5GA1UECgwE  
QkpDQTENMA5GA1UECwwEQkpDQTEXMBUGA1UEAwO  
QmVpamluZyBTTTIgQ0EwHhcNMjQwNzExMTYwMDAwWhc  
NMjkwNzEyMTU1OTU5WjB8MRwwGgYDVQQpDBNOOTEx  
MTAxMDgwODk2Mzg5OVhRMU8wTQYDVQQDDDEZCSkdYQ  
OS4reWbveaLm+agh+WFrOWFseacjeWKoeW5s+WP5OaciemZk  
OWFrOWPuEBOTjkkMTEwMTA4MDg5NjM4MTIYUUAxMQs  
wCQYDVQQGDAJDTjBZMBMGBYqGSM49AgEGCCqBHM9V  
AYItA0IABGVH8jpDsD9hcvhowe9wSPaSrKliPFGXj/f/c3D54Iye  
9dG66gR+VThi7hHXBjDZHGjsYv8J2Fn1ZsXMRHFi1cmjggL9  
MIIC+TAfBgNVHSMEGDAWgBQf5s/Uj8UiKpdKKYoV5xbJkj
```

TEtjAdBgNVHQ4EFgQU4phbBnAz3uclS2Tqn+10eKyktQowgZ8
GA1UdHwSBlzCBIDBhoF+gXaRbMFkxCzAJBgNVBAYTAkN
OMQ0wCwYDVQQKDARCSkNBMQ0wCwYDVQQLDARCSk
NBMRcwFQYDVQQDDA5CZWlqaW5nIFNNMiBDQTETMBE
GA1UEAxMKY2EyMWNybdQxMTAvoC2gK4YpaHR0cDovL2
NybC5iamNhLm9yZy5jbi9jcmwvY2EyMWNybdQxMS5jcmwwJ
QYKKoEchu8yAgEBAQQXDBVOIE45MTExMDEwODA4OTY
zODE5WFewYAYIKwYBBQUHAQE EVDBSMCMGCCsGAQU
FBzABhhdPQ1NQOi8vb2NzcC5iamNhLm9yZy5jbjArBggrBgEF
BQcwAoYfaHR0cDovL2NybC5iamNhLm9yZy5jbi9jYWlzc3Vlcj
BABgNVHSAEOTA3MDUGCSqBHIbvMgICATAoMCYGCCsG
AQUFBwIBFhpodHRwOi8vd3d3LmJqY2Eub3JnLmNuL2Nwcza
RBglghkgBhvhCAQEEBAMCAP8wIwYKKoEchu8yAgEBCAQ
VDBNOOTExMTAxMDgwODk2MzgxOVhRMCUGCiqBHIbvM
gIBAgIEFwwVTiBOOTExMTAxMDgwODk2MzgxOVhRMB8G
CiqBHIbvMgIBAQQ4EEQwPMTAyMDgwMDQ1NDgyMjI5MCU
GCiqBHIbvMgIBAQQEFwwVTiBOOTExMTAxMDgwODk2Mz
gxOVhRMBQGCIqBHIbvMgIBAwEEBgwEMTAyODAuBgoqgR
yG7zICAQEXBCAMHjJAMjE1MDA5TiAwTjkxMTEwMTA4M
Dg5NjM4MTIYUTAUBgoqgRyG7zICAQEaBAYMBDEwMjgwI
QYIKoEc0BQEAQQEFQwTTjkxMTEwMTA4MDg5NjM4MTIY


```
UTAUBgoqgRyG7zICAQEeBAYMBDMwMzgwDgYDVR0PAQ  
H/BAQDAgbAMAOGCCqBHM9VAYN1A0kAMEYCIQDr0miP  
n45tMr65cmY9k+VMg/gwzF8OkBd6bhJhv0meHwIhAPIFbBWV  
H7Ua95vepGf64G1pXgvI4+uvoIPenlVnmS1n";
```

```
        X509Certificate readRootCertificate =  
readCertificate(caCertBase64);
```

```
        //验证证书有效性，是否为根证书签发的
```

```
        boolean isValid =
```

```
verifySubCertificateValid(readRootCertificate,  
readRootCertificatesub);
```

```
        System.out.println("该子证书是否由根证书签发 " +  
isValid);
```

```
    }
```

```
/**
```

```
 * 验证证书是否由根证书签发
```

```
 *
```

```
 * @param rootCert 根证书
```

```
 * @param subCert 子证书
```

```
 */
```

```
private static boolean
```

```
verifySubCertificateValid(X509Certificate rootCert,
```

```

X509Certificate subCert) {
    try {
        subCert.verify(rootCert.getPublicKey());
        return true;
    } catch (Exception e) {
        e.printStackTrace();
        return false;
    }
}

/**
 * 读取证书文件
 */

private static X509Certificate readCertificate(String
subCertBase64) throws IOException, CertificateException,
NoSuchProviderException {
    byte[] certBytes = Base64.decode(subCertBase64);
    CertificateFactory certFactory =
CertificateFactory.getInstance("X.509", "BC");
    return (X509Certificate)
certFactory.generateCertificate(new
ByteArrayInputStream(certBytes));

```

```

    }

    // 公钥加密
    public static String encrypt(String publicKey, String
plaintext) {
        SM2 sm2 = new SM2(null, publicKey);

        return sm2.encryptHex(plaintext.getBytes(),
KeyType.PublicKey);
    }

    // 私钥解密
    public static String decrypt(String privateKey, String
ciphertext) {
        SM2 sm2 = new SM2(privateKey, null);
        return sm2.decryptStr(ciphertext, KeyType.PrivateKey);
        //    return new
String(sm2.decrypt(Base64.decode(ciphertext),
KeyType.PrivateKey));
    }

```

```
}
```

引入以下依赖：

XML

```
<!-- BouncyCastle 加密库，支持 SM2 算法 -->
```

```
<dependency>
```

```
<groupId>org.bouncycastle</groupId>
```

```
<artifactId>bcprov-jdk15on</artifactId>
```

```
<version>1.65</version>
```

```
</dependency>
```

```
<!-- hutool 工具类 -->
```

```
<dependency>
```

```
<groupId>cn.hutool</groupId>
```

```
<artifactId>hutool-all</artifactId>
```

```
<version>5.8.25</version>
```

```
</dependency>
```

4.2.附录 2:CA 机构编码

CA 机构编码	CA 机构名称	CA 机构简称	统一社会信用代码
【010001】	中金金融认证中心有限公司	CFCA	91110000759626025U
【010002】	北京天威诚信电子商务服务有限公司	天威诚信 CA	911101088020176135
【010003】	北京数字认证股份有限公司	北京 CA	91110108722619411A
【110004】	颐信科技有限公司	颐信 CA	911101087404063055
【110005】	北京国富安电子商务安全认证有限公司	国富安 CA	91110302633713895D
【110006】	联通智慧安全科技有限公司	联通 CA	91110302665605522J
【010007】	北京中认环宇信息安全技术有限公司	中认 CA	911101065769084175
【110008】	中铁信弘远（北京）软件科技有限责任公司	中铁 CA	9111010870022358XF
【110009】	北京世纪速码信息科技有限公司	CSCA	91110108061266945F
【110010】	农信银资金清算中心有限责任公司		911100007889504067
【110011】	中国电力科学研究院有限公司		91110000400007201W
【110012】	泰尔认证中心有限公司	泰尔 CA	91110102746706317J

CA 机构编 码	CA 机构名称	CA 机构 简称	统一社会信用代 码
【110013】	国汽（北京）智能网联 汽车研究院有限公司	国汽智联 CA	91110302MA01A UHX2M
【110014】	北京群盾科技有限公司	群盾 CA	91110108MA003H F80R
【440001】	广东省电子商务认证有 限公司	网证通 CA	914400007250611 93F
【440002】	数安时代科技股份有限 公司	GDCA	914406007470989 58H
【020003】	深圳市电子商务安全证 书管理有限公司	深圳 CA	914403007230387 58B
【440004】	卓望数码技术（深圳） 有限公司	卓望 CA	914403007230493 18F
【440005】	沃通电子认证服务有限 公司	沃通 CA	914403007362667 09U
【320001】	江苏省国信数字科技有 限公司	国信 CA	91320000MA1UQ C6253
【320002】	江苏智慧数字认证有限 公司	江苏智慧 CA	91320114MA1Q4 RMM8Y
【320003】	江苏国密数字认证有限 公司	国密 CA	91320114MA1WK PF869
【320004】	南京数字认证有限公司	南京 CA	91320191MA1WL 8B040
【320005】	江苏公众数科数字科技 有限公司		91320111MA22J1 RC9G
【500001】	东方中讯数字证书认证	东方中讯	915001087093673

CA 机构编 码	CA 机构名称	CA 机构 简称	统一社会信用代 码
	有限公司		36L
【500002】	重庆程远未来电子商务 服务有限公司	程远未来	915000003223922 97X
【500003】	大陆云盾电子认证服务 有限公司	大陆云盾	91500108MA5YQ 7P665
【430001】	湖南省数字认证服务中 心有限公司	湖南 CA	914300007580308 95K
【430002】	东方新诚信数字认证中 心有限公司	DFCA	914301007225709 294
【430003】	苏博云科数字认证有限 公司	苏博云科 CA	914301007607456 972
【120001】	天津市滨海数字认证有 限公司	滨海 CA	911201163408731 074
【120002】	天津数字认证有限公司 (曾用名: 天津市中环 认证服务有限公司)	天津 CA	91120103MA06B C2X8K
【120003】	中汽数据(天津)有限 公司		91120111MA05UF 8E12
【021001】	上海市数字证书认证中 心有限公司	上海 CA	913100006312912 89X
【310002】	亚数信息科技(上海) 有限公司	亚洲诚信	913101040660431 939
【370001】	山东省数字证书认证管 理有限公司	山东 CA	913700007266934 67T
【370002】	山东彗信认证服务有限	彗信 CA	913701000761767

CA 机构编 码	CA 机构名称	CA 机构 简称	统一社会信用代 码
	公司（原山东云海）		662
【410001】	华测电子认证有限责任 公司	华测 CA	914101007457903 34N
【410002】	河南省信息化集团有限 公司（曾用名：河南省 信息化发展有限公司）	信安 CA	914100006921523 38A
【330001】	浙江省数字安全证书管 理有限公司	浙江 CA	913300007458310 011
【530001】	云南省数字证书认证中 心有限公司	云南 CA	915301005746867 42H
【650001】	新疆数字证书认证中心 （有限公司）	新疆 CA	916500007576560 875
【510001】	四川省数字证书认证管 理中心有限公司	四川 CA	915101006696506 706
【029001】	陕西省数字证书认证中 心股份有限公司	陕西 CA	916100007552141 62L
【140001】	山西省数字证书认证中 心（有限公司）	山西 CA	911401007319116 5X3
【640001】	西部安全认证中心有限 责任公司	西部 CA	916411007150157 08T
【150001】	内蒙古网信电子认证有 限责任公司	腾讯云 CA	911501025706334 83W
【210001】	辽宁数字证书认证管理 有限公司	辽宁 CA	912100007591174 82Q
【360001】	江西省数字证书有限公	江西 CA	913600007633837

CA 机构编 码	CA 机构名称	CA 机构 简称	统一社会信用代 码
	司		406
【220001】	吉林省安信电子认证服 务有限公司	安信 CA	912201017359355 01K
【420001】	湖北省数字证书认证管 理中心有限公司	湖北 CA	914200007146559 654
【230001】	黑龙江省数字证书认证 有限公司	黑龙江 CA	912301107950171 47W
【130001】	河北省电子认证有限公 司	河北 CA	911300007329144 9XE
【460001】	海南省信安电子认证有 限公司	海南信安 CA	91460100MA5T46 WB4J
【520001】	贵州省电子证书有限公 司	贵州 CA	915201007753135 128
【450001】	广西壮族自治区数字证 书认证中心有限公司	广西 CA	914500007479504 85J
【350001】	福建省数字安全证书管 理有限公司	福建 CA	913500007336001 349
【340001】	安徽省电子认证管理中 心有限责任公司	安徽 CA	913400007790764 975