iSignature_OFD_API

V2.0.0.98

中 间 件 技 术 白 皮 书

地址： 江西南昌高新技术开发区火炬大街 579 号绿悦科技大厦 15 楼
邮编： 330096
网址： http://www.kinggrid.com
电话： *0791-82221588*
服务： *400-6776-800*

目录

1 引言

1.1. 模块概述

产品模块为可选件，按需选购提供。

1.1.1 电子签章

普通印章、骑缝章（包括双侧对开）、删除印章、获取印章信息、获取证书信息、验证签名有效性

1.2. 开发平台

开发平台：windows 7、Eclipse 3.7

编程语言：java

编译版本：JDK6

1.3. 运行环境

软件环境：JDK1.5 及以上的版本

操作系统：windows、linux、unix，支持 64 位操作系统

2 接口说明

类名：

com.kinggrid.ofd. KGOfdHummer

2.1 接口列表

名称	描述	备注
createInstance	创建编辑文档实例	
addExecute	添加任务接口	
doSignature	执行任务，添加数字签名	
doExecute	执行任务	
setMultiUserSign	是否开启前台 *key* 签章的多人会签模式	
DigitSign	*SM2* 签名接口	
VerifySign	*SM2* 验签接口	
getKGOfdHummerUtils	获取辅助类	
getPageOfNumber	获取文档总页数	

2.2 接口详细说明

2.2.1 createInstance

1. ***public static KGOfdHummer createInstance(String fileName,byte[] ownerPassword, String outPath)***

功能说明：

创建文档编辑实例。

参数说明：

fileName：需要盖章的文档路径。

ownerPassword：文档打开密码，没有传 *null*。

outPath：签章后的文档。

返回值：

KGOfdHummer 实例。

2.2.2 addExecute

1. ***public void addExecute(KGExecute execute)***

功能说明：

添加任务接口。

参数说明：

无。

返回值：

void

2.2.3 doSignature

1. ***public void doSignature(String signData, String signDirPath,String outPath)***

功能说明：

执行任务，前台 *key* 签章添加签名值。

参数说明：

signData：签名值。

signDirPath：添加签名值的 *sign* 文件夹路径。

outPath：签章后的文档。

返回值：

void

2.2.4 doExecute

1. ***public void doExecute()***

功能说明：

执行任务。

参数说明：

无。

返回值：

void

2.2.5 setMultiUserSign

2. ***public void setMultiUserSign (boolean multiUserSign)***

功能说明：

设置是否开启前台 *Key* 的多人会签模式。

参数说明：

无。

返回值：

void

2.2.6 DigitSign

3. ***public String DigitSign(byte[] originalText, KGSignType signType)***

功能说明：

SM2 签名接口。

参数说明：

originalText：原文。

signType：签名算法。

返回值：

签名值

2.2.7 VerifySign

4. ***public boolean VerifySign (byte[] originalText, String signData, KGSignType signType)***

功能说明：

SM2 验签接口。

参数说明：

originalText：原文。

signData：签名值。

signType：签名算法

返回值：

验签结果

2.2.8 getKGOfdHummerUtils

5. ***public KGOfdHummerUtils getKGOfdHummerUtils()***

功能说明：

获取辅助类。

参数说明：

无。

返回值：

KGOfdHummerUtils：辅助类

2.2.9 getPageOfNumber

6. ***public int getPageOfNumber()***

功能说明：

获取文档总页数。

参数说明：

无。

返回值：

总页数

3 任务接口说明

3.1 电子签章接口

3.1.1 添加电子签章

类名：*com.kinggrid.ofd.executes.OfdElectronicSeal4KG*

构造方法：

1. ***OfdElectronicSeal4KG (String url, String keySN, String password,String signName)***

   签章服务器对接

   url：签章服务器地址 *http://127.0.0.1:8089/iSignatureServer/OfficeServer.jsp*

   keySN：密钥盘序列号。

   password：用户密码，*CA0* 版才会验证密码。

   signName：印章名称。

2. ***OfdElectronicSeal4KG (String url , String keySN, String password,int index)***

   签章服务器对接

   url：签章服务器地址 *http://127.0.0.1:8089/iSignatureServer/OfficeServer.jsp*

   keySN：密钥盘序列号。

   password：用户密码，*CA0* 版才会验证密码。

   index：当前印章序号，从 *0* 开始。

方法：

1. ***public void setPagen(int pagen)***

   设置需要操作的页码。

   pagen：页码，*0* 表示所有页

2. ***public void setType (KGServerTypeEnum type)***

   设置签章服务器的类型

   type：服务器类型，*AUTO_GM*

3. ***public void setVerifySeal(boolean verifySeal)***

   设置是否进行盖章前对印章数据的验证

4. ***public void setSealMsg(byte[] sealData)***

   设置印章数据

   sealData：印章数据

5. ***public void setXY(float x, float y)***

   坐标定位

   OFD 文档左上角为原点，设置的坐标为印章左上角的坐标

6. ***public void setTree (String semanticTree)***

   语义树定位，所有匹配的语义树都会加盖印章

7. ***public void setTree (String semanticTree,boolean onceFindReturn)***

   语义树定位

   semanticTree : 需要定位的语义树内容

   treeOnceFindReturn :

   true : 只在第一个匹配的语义树上加盖印章

   false : 所有匹配的语义树都加盖印章

8. ***public void setText (String text)***

   文本定位，所有匹配的文本都会加盖印章

9. ***public void setText (String text, boolean textOnceFindReturn)***

   文本定位

   text : 需要定位的文本内容

   textOnceFindReturn :

   true : 只在第一个匹配的文本上加盖印章

   false : 所有匹配的文本都加盖印章

10. ***public void setTextOffset(float offsetX, float offsetY)***

    文本定位时，设置相对文本位置的偏移量。

    OFD 左上角为坐标原点。

7. ***public void qfz(int number,KGQfzModeEnum qfzMode,int yDistanceOrigin)***

   骑缝章

   number : *number* 页均分一个章，循环分配。模（余数：*mod*）等于 *1*，后面 *number+1* 页均分一个章；否则 *mod* 页均分一个章。*2<=number<=32*。

   qfzMode: *KGQfzModeEnum.ALLPAGE* 所有页

   KGQfzModeEnum.ODDPAGE 奇数页

   KGQfzModeEnum.EVENPAGE 偶数页

   yDistanceOrigin: 印章距离顶部的距离

8. ***public void qfzBilateralOff(int yDistanceOrigin)***

   骑缝章，双侧对开

9. ***public void saveLog(String documentID,String documentName,String logMemo, String extparam1,String extparam2)***

   记录日志，必须是印章来着签章服务器，才能成功保存日志。

documentID：文档编码

documentName：文档名称

logMemo：日志描述

extparam1：扩展项 *1*

extparam2：扩展项 *2*

10. ***public void setInfomation(SignatureInter inter)***

设置通过“签章服务器”或者“前台 *Key*”对文档进行签章

1) *SignatureInterByServer(OfdElectronicSeal4KG ofdElectronicSeal4KG, String url)*：从签章服务器获取印章通过加密机对文档进行签章

所需参数：

ofdElectronicSeal4KG：对接签章服务器的印章类

url ：签章服务器地址

2) *SignatureInterByServer (OfdElectronicSeal4KG ofdElectronicSeal4KG, String url, String keysn)*：从签章服务器获取印章通过签章服务器对接密管系统对文档进行签章

所需参数：

ofdElectronicSeal4KG：对接签章服务器的印章类

url ：签章服务器地址

keysn ：密钥盘序列号

3) *SignatureInterByKey (String cer, OfdElectronicSeal4KG ofdElectronicSeal4KG)*：通过前台 *key* 对文档进行签章

所需参数：

cer ：证书数据

ofdElectronicSeal4KG：对接签章服务器的印章类

11. ***public void setLockSign (boolean isLockSign)***

设置为是否为锁定签名。

3.1.2 添加电子签章（对接应用系统加盖政务印章）

类名：*com.kinggrid.ofd.executes. OfdElectronicSeal4GB*

构造方法：

1. ***OfdElectronicSeal4GB (String url, String salt, String appid, String esid)***

应用系统对接

url：应用系统部署地址 *http://192.168.0.184:8081/iSignatureStamp*

Salt：盐值

appid: 授权码。

esid：印章编码。

3.1.3 电子签章信息

类名：*com.kinggrid.ofd.executes.OfdSignDetails*

属性：无

1. **方法**：***public List<SignInfo> getSignInfos ()***

   签章信息列表

说明：无

3.1.4 删除电子签章

类名：*com.kinggrid.ofd.executes. DeleteSignature*

属性：无

1. **方法**：***public void setDeleteLastOneSeal(boolean deleteLastOneSeal)***

   设置是否删除最后一个印章

说明：删除全部电子签章

4 任务接口实例

4.1 电子签章实例

4.1.1 添加电子签章

```
public void testSignatureByServer() {
    KGOfdHummer hummer = null;
    try {
        hummer = KGOfdHummer.createInstance("D:/ofd/test.ofd", null, "D:/ofd/1.ofd");
        InputStream cer = new FileInputStream(new File("D:/ofd/servercert.cer"));
        //设置开启多人会签模式(只对前台插 Key 模式签章有效)
        //hummer.setMultiUserSign(true);
        String url = "http://171.34.78.70:8889/iSignatureServer/OfficeServer.jsp";
        OfdElectronicSeal4KG ofdElectronicSeal4KG = new
OfdElectronicSeal4KG(url,"test001",null,0);
        ofdElectronicSeal4KG.setType(KGServerTypeEnum.AUTO_GM);
        ofdElectronicSeal4KG.setVerifySeal(false);
        SignatureInterByServer signatureInterByServer = new
SignatureInterByServer(ofdElectronicSeal4KG,url);
    //  signatureInterByServer.setHashType(KGHashType.HASH_TYPE_SHA1);
        signatureInterByServer.setCer(cer);
        ofdElectronicSeal4KG.setInfomation(signatureInterByServer);
    //  ofdElectronicSeal4KG.setTree("定义");
        ofdElectronicSeal4KG.setXY(120,220);
    //  ofdElectronicSeal4KG.setPagen(1);
    //  ofdElectronicSeal4KG.qfz(2,KGQfzModeEnum.ALLPAGE, 100);
    //  ofdElectronicSeal4KG.qfzBilateralOff(60);
    //  ofdElectronicSeal4KG.setTree("抄送机关");
    //  ofdElectronicSeal4KG.setText("注册资本", false);
    //  ofdElectronicSeal4KG.setTextOffset(10, 10);
    //  ofdElectronicSeal4KG.setLockSign(true);
    //  ofdElectronicSeal4KG.saveLog("documentID", "documentName", "logMemo",
```

```
"extparam1", "extparam2");
        hummer.addExecute(ofdElectronicSeal4KG);

    // OfdElectronicSeal4KG ofdElectronicSeal4KG1 = new
OfdElectronicSeal4KG(url,"test001",null,0);
    // ofdElectronicSeal4KG1.setType(KGServerTypeEnum.AUTO_GM);
    // SignatureInterByServer signatureInterByServer1 = new
SignatureInterByServer(ofdElectronicSeal4KG1,url);
    // signatureInterByServer1.setCer(cer);
        //对接签章服务器，服务器对接加密机系统时设置
    // signatureInterByServer.setKeysn("1233335555KGCS6666");
    // ofdElectronicSeal4KG1.setInfomation(signatureInterByServer1);
    // ofdElectronicSeal4KG1.setXY(50,20);
    // ofdElectronicSeal4KG1.setPagen(1);
    // ofdElectronicSeal4KG1.setTree("抄送机关",false);
    // hummer.addExecute(ofdElectronicSeal4KG1);

        hummer.doExecute();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

4.1.2 添加电子签章（签章服务器对接密管系统加盖国密印章）

```
    public void testSignByMG() throws IOException, ZipException, DocumentException {
        String url = "http://192.168.0.184:8082/iSignatureServer/OfficeServer.jsp";
        String keysn = "test002";

        KGOfdHummer hummer = KGOfdHummer.createInstance("D:/ofd/金格科技〔2006〕6
号.ofd", null, "D:/ofd/1.ofd");
        OfdElectronicSeal4KG ofdElectronicSeal4KG = new
    OfdElectronicSeal4KG(url,keysn,"123456",0);
        ofdElectronicSeal4KG.setType(KGServerTypeEnum.AUTO_GM);
        SignatureInterByServer signatureInterByServer = new
```

```
    SignatureInterByServer(ofdElectronicSeal4KG, url, keysn);
        ofdElectronicSeal4KG.setInfomation(signatureInterByServer);
        ofdElectronicSeal4KG.setXY(10, 10);
        ofdElectronicSeal4KG.setPagen(1);

        hummer.addExecute(ofdElectronicSeal4KG);
        hummer.doExecute();
    }
```

4.1.3 添加电子签章（对接应用系统加盖政务印章）

```
public void testSign() {
    Security.addProvider(new BouncyCastleProvider());
    KGOfdHummer hummer = null;
    try {
        hummer = KGOfdHummer.createInstance("D:/ofd/金格科技〔2006〕6号.ofd",
null,"D:/ofd/1.ofd");
        String stampSystemUrl = "http://192.168.0.184:8081/iSignatureStamp";
        String appid = "340BE82F1C3949D39796E7A2AA76B4D4";
        String esid = "36000000000155";
        String salt = "2ad57f1d158b4a63ac248c423ef09cd8";
        OfdElectronicSeal4GB ofdElectronicSeal4GB = new
OfdElectronicSeal4GB(stampSystemUrl, salt, appid, esid);
        SignatureInterByGBImpl signatureInterByGBImpl = new
SignatureInterByGBImpl(stampSystemUrl, salt, appid, esid, "1", "title");
        ofdElectronicSeal4GB.setInfomation(signatureInterByGBImpl);
        ofdElectronicSeal4GB.setXY(100,100);
        hummer.addExecute(ofdElectronicSeal4GB);
        hummer.doExecute();
} catch (Exception e) {
        e.printStackTrace();
    }
}
```

4.1.4 电子签章信息 & 验证签名

```
public void testSealDetails() {
    KGOfdHummer hummer = null;
        try {
            hummer = KGOfdHummer.createInstance("D:/ofd/test.ofd", null);
            OfdSignDetails signDetails = new OfdSignDetails();
            hummer.addExecute(signDetails);
            hummer.doExecute();
            List<SignInfo> signInfos = signDetails.getSignInfos();
            for (SignInfo signInfo : signInfos) {
                System.out.println("\n\n 序号：" + signInfo.getSealInfo().getIndex());
                System.out.println("------印章信息------");
                System.out.println("厂商 ID：" + signInfo.getSealInfo().getSealVid());
                System.out.println("印章标识:" + signInfo.getSealInfo().getSealID());
                System.out.println("印章类型：" + signInfo.getSealInfo().getType());
                System.out.println("印章名称：" + signInfo.getSealInfo().getSealName());
                System.out.println("起始时间：" + signInfo.getSealInfo().getValidStart());
                System.out.println("结束时间：" + signInfo.getSealInfo().getValidEnd());
                System.out.println("印章签名值" + signInfo.getSealInfo().getSignData());
                System.out.println("印章制作时间：" +
signInfo.getSealInfo().getCreateDate());
                System.out.println("签章时间：" + signInfo.getSealInfo().getSignDate());
                System.out.println("制章人：" + signInfo.getSealInfo().getUserName());
                System.out.println("印章位置" + signInfo.getSealInfo().getLlx() + "-" +
signInfo.getSealInfo().getLly() + "-" + signInfo.getSealInfo().getUrx() + "-" +
signInfo.getSealInfo().getUry());
                System.out.println("盖章页码：" +
Arrays.toString(signInfo.getSealInfo().getPagen()));
                System.out.println("------印章信息------");
                System.out.println("------证书信息------");
                System.out.println("证书版本：" +
signInfo.getCertificateStructure().getVersion());
                System.out.println("序列号：" +
signInfo.getCertificateStructure().getSerialNumber().getValue().toString(16));
```

```
                System.out.println("算法标识：" +
signInfo.getCertificateStructure().getSignatureAlgorithm().getObjectId().getId());
                System.out.println("签发者：" +
signInfo.getCertificateStructure().getIssuer());
                System.out.println("开始时间：" +
signInfo.getCertificateStructure().getStartDate().getTime());
                System.out.println("结束时间：" +
signInfo.getCertificateStructure().getEndDate().getTime());
                System.out.println("主体名：" +
signInfo.getCertificateStructure().getSubject());
                SubjectPublicKeyInfo pukinfo =
signInfo.getCertificateStructure().getSubjectPublicKeyInfo();
                System.out.println("标识符：" +
pukinfo.getAlgorithmId().getObjectId().getId());
                byte[] byPuk = pukinfo.getPublicKeyData().getBytes();
                String strPuk = new String(Hex.encode(byPuk));
                System.out.println("公钥值：" + strPuk);
                System.out.println("------证书信息------\n");

                System.out.println("验证证书结果：" + signInfo.cerValid());
                System.out.println("签证签名结果：" + signInfo.verifySign(signDetails));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

4.1.5 删除电子签章

```
public void testDelSignatures(){
    KGOfdHummer hummer = null;
    try {
        hummer = KGOfdHummer.createInstance("D:/ofd/test.ofd", null, "D:/ofd/1.ofd");
        DeleteSignature deleteSignature = new DeleteSignature();
        hummer.addExecute(deleteSignature);
        hummer.doExecute();
```

```
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

4.1.6 骑缝章及双侧

```
public void testSigntureQfz() {
    KGOfdHummer hummer = null;
    try {
        hummer = KGOfdHummer.createInstance("D:/ofd/test.ofd", null, "D:/ofd/1.ofd");
        File cer = new File("D:/ofd/servercert.cer");
        String url = "http://171.34.78.70:8899/iSignatureServer/OfficeServer.jsp";
        OfdElectronicSeal4KG ofdElectronicSeal4KG = new
OfdElectronicSeal4KG(url,"0672031308030317",null,0);
        ofdElectronicSeal4KG.setInfomation(new
SignatureInterByServer(ofdElectronicSeal4KG,cer,url));
        ofdElectronicSeal4KG.qfz(2,KGQfzModeEnum.ALLPAGE, 100);
        ofdElectronicSeal4KG.qfzBilateralOff(60);
        hummer.addExecute(ofdElectronicSeal4KG);
        hummer.doExecute();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

4.2 批注实例

4.2.1 插入批注实例

合并阅读器 *KGReader* 导出的批注信息。

```
public void testAnnot() {
        try {
            OfdAnnotations ofdAnnot = new OfdAnnotations();
            String data = "Annotation Information";
```

```
            ofdAnnot.addAnnotationByStr("D:/ofd/test.ofd", data);
        }catch (Exception e) {
            e.printStackTrace();
        }
    }
```

4.2.2 删除批注实例

```
public void testDeleteAnnot() {
        KGOfdHummer hummer = null;
        try {
            hummer = KGOfdHummer.createInstance("D:/ofd/test.ofd", null,
"D:/ofd/delete.ofd");
            DeleteAnnotations deleteAnnotations = new DeleteAnnotations();
            deleteAnnotations.setCreator("国泰");
//          deleteAnnotations.setSubtype(AnnoteType.Pencil);

            hummer.addExecute(deleteAnnotations);
            hummer.doExecute();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

5 其他说明

5.1 KGOfdHummerUtils 辅助类说明

类名：*com.kgofd.ofd. KGPdfHummerUtils*

方法：

1. ***public List<Position> getPositionByText(int pageNumber,String text,boolean once)***

   说明：查找某页文本所在的位置，返回位置

   参数：

   pageNumber：页数。

   text：文本

   once：是否找到对应文本第一个位置就返回。

   返回：

   矩形框的位置

6 文档声明

本文档内容改动及版本更新将不再另行通知。本文档的范例中使用的人名、公司名和数据如果没有特别指明，均属虚构。对于本文档、及本文档涉及的技术和产品，江西金格科技股份有限公司拥有其专利、商标、著作权或其它知识产权，除非得到江西金格科技股份有限公司的书面许可，本文档不授予这些专利、商标、著作权或其它知识产权的许可。

- *Kinggrid*、*iWebOffice*、*iSignature*和*DBPacket*是江西金格科技股份有限公司的商标。
- 其它标牌和产品名称是其各自公司的商标或注册商标。
- 本文档最后更新时间：*2019.04.10*。

（完）