无纸化电子签章系统 API 手册

Version 3.6

目 录

1 引言

1.1 概述

此文档是对 CFCA 无纸化电子印章系统接口的使用说明。

1.2 开发平台及编程语言

- 开发平台
  Windows 7
- 编程语言
  JAVA

1.3 名词解释

- 数字证书

  数字证书是各类实体（持卡人/个人、商户/企业、网关/银行等），在网上进行信息交流及商务活动的身份证明，在电子交易的各个环节，交易的各方都需验证对方证书的有效性，从而解决相互间的信任问题。简单的说，数字证书是一段包含用户身份信息、用户公钥信息以及身份验证机构数字签名的数据。身份验证机构的数字签名可以确保证书信息的真实性，证书格式及证书内容遵循 X.509 标准。

  从证书的一般用途来看，数字证书可分为签名证书和加密证书。签名证书主要用于对用户信息进行签名，以保证信息的不可否认性；加密证书主要用于对用户传送信息进行加密，以保证信息的真实性和完整性。

- 电子印章
  电子印章是电子签名一种表现形式，利用图像处理技术将电子签名操作转化为与纸质文件盖章操作相同的可视效果，同时利用电子签名技术保障电子信息的真实性和完整性以及签名人的不可否认性。
- 数字签名

  在进行数字签名过程中，发信者使用自己的私钥，通过非对称密码算法，对待发数据的数字摘要（哈希值）进行加密，从而得到一段信息称为数字签名（Sign）。

- 证书吊销列表

  证书吊销列表 CRL 是一种包含吊销的证书列表的签名数据结构。CRL 是证书吊销状态的公布形式，CRL 就像信用卡的黑名单，它通知其他证书用户某些电子证书不再有效。

2 无纸化基本接口

2.1 PDF 自动化合成模板业务数据接口

public byte[] **synthesizeAutoTemplate**(String templateCode, String savedPdfId, String fieldBeanListXml，String textBeanListXml, String imageBeanListXml, String operatorCode, String channelCode) throws Exception

使用场景：合成业务数据到模板生成单据，业务数据包括文本、图片

入参：

templateCode：模板号，支持传入多个模版号时返回合并后的 pdf，比如：“tmp001,tmp002”

savedPdfId：保存到临时表中的 pdfId，符合唯一约束，可为空

fieldBeanListXml: 需要合成到 pdf 文本域的业务数据，可为空，有两种格式，第一种格式如下：

```
<FieldList hashAlg="sha1" savedBizXmlId="" savedTimeIncrement="">
    <Field>
        <FieldId></FieldId>必填
        <FieldName></ FieldName>字段名称，可为空
        <FieldValue></FieldValue>可以为空
        <FieldType></ FieldType> FieldType：text,checkbox
        <FieldValueHash></FieldValueHash>可为空
        <TemplateIndex> </TemplateIndex>//模板序号，取值：1、2、3...，不为空且不为 0 时合成到指定模板号，为空或 0 时所有字段合成到所有模板，此字段只适应此方法
    </Field>
<FieldList>
```

第二种格式：bizXmlId，即之前保存的 savedBizXmlId

参数解释：savedBizXmlId：报文保存的 ID，下同

savedTimeIncrement：报文保存的时间增量（分钟），不可以是负数，下同

textBeanListXml: 需要合成到 pdf 内容中的业务数据，可为空，有两种格式，第一种格式如下：

```
<TextList hashAlg="sha1" savedBizXmlId="" savedTimeIncrement="">
    <Text>
        <TextValue></ TextValue >必填
        <TextValueHash></ TextValueHash>可为空
        <TextFontSize></ TextFontSize>字体大小，比如：10、11、12
        <TextColor></ TextFontSize>FFFFFF、000000
        <Type></Type>类型，2：坐标；3：关键字，必填
        <PageNo>1</PageNo> 页码，必填
        <LX>100</LX> 必填
        <LY>100</LY> 必填
```

<Keyword></ Keyword>关键字，必填
< LocationStyle></ LocationStyle>关键字风格，C、U、D、L、R,必填
< OffsetX></ OffsetX>X 轴偏移量，可为空
< OffsetY></ OffsetY> X 轴偏移量，可为空
<TemplateIndex> </TemplateIndex>//*模板序号，同上*
</Text>
<TextList>
第二种格式：bizXmlId，即之前保存的 savedBizXmlId
imageBeanListXml: 需要合成到 pdf 的图片数据，可为空，有两种格式，第一种格式如下：
<ImageList hashAlg="sha1" savedBizXmlId="" savedTimeIncrement="">
<Image>可以没有该节点
< ImageValue ></ ImageValue >图片 base64 编码，必填
<ImageWidth></ ImageWidth >图片宽度，必填
<ImageHeight></ ImageHeight >图片高度，必填
<ImageHash></ ImageHash >可为空
<Type></Type>类型，2：坐标；3：关键字，必填
<PageNo>1</PageNo> 页码，必填
<LX>100</LX> 必填
<LY>100</LY> 必填
<Keyword></ Keyword>关键字，必填
< LocationStyle></ LocationStyle>关键字风格，C、U、D、L、R,必填
< OffsetX></ OffsetX>X 轴偏移量，可为空
< OffsetY></ OffsetY> X 轴偏移量，可为空
<TemplateIndex> </TemplateIndex>//*模板序号，同上*
</Image>
<QRCodeXml>可为空，也可以是多个
<QRCodeContent></ QRCodeContent>二维码内容，必填
<Width></ Width>二维码宽度默认，必填
<Height></ Height>二维码高度默认，必填
<PageNo>1</PageNo> 页码，必填
<LX></LX>二维码存放在 pdf 中的位置，必填
<LY></LY>二维码存放在 pdf 中的位置，必填
<TemplateIndex> </TemplateIndex>//*模板序号，同上*
</QRCodeXml>
</ImageList >
第二种格式：bizXmlId，即之前保存的 savedBizXmlId
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空

正常返回：

合成后的 pdf 文件

异常返回：

<Error>
<ErrorCode>errorCode</ErrorCode>

<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

2.2 PDF 自动化合成业务数据接口

public byte[] **synthesizeAutoBusinessPdf**(byte[] pdf, String savedPdfId, String fieldBeanListXml， String textBeanListXml, String imageBeanListXml, String operatorCode, String channelCode) throws Exception
使用场景：合成业务数据到 pdf 中，业务数据包括文本域、文本和图片
入参：
pdf：pdf 文件数据或者是 pdfId（pdfId 表示之前保存的 savedPdfId）
savedPdfId：临时保存的 pdfId，可以为空
fieldBeanListXml：需要合成到 pdf 文本域的业务数据或者是 bizXmlId（同上），可为空
textBeanListXml：需要合成到 pdf 内容中的业务数据或者是 bizXmlId（同上），可为空
imageBeanListXml：图片列表或者是 bizXmlId（同上），可为空
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空

2.3 PDF 自动化合成多媒体数据接口

public byte[] **synthesizeAutoMultiPdf**(byte[] pdf, String savedPdfId, String multiData，String operatorCode, String channelCode) throws Exception
使用场景：合成多媒体数据到 pdf 中，多媒体数据包括录音、图片等
入参：
pdf：pdf 文件数据或者是 pdfId（pdfId 表示之前保存的 savedPdfId）
savedPdfId：临时保存的 pdfId，可以为空
multiData: 需要合成到 pdf 的多媒体数据，multiData 为 Json 格式如下：
{ {“type”:”0”,”fileName”:”fileNameValue”,”fileData”:”fileDataBase64String”},{ },… }
type 取值：1=注释方式；2=附件方式；0：不放到 pdf 文档（需要客户自己保存）
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空
正常返回：
合成后的 pdf 文件
异常返回：
<Error>
<ErrorCode>errorCode</ErrorCode>

<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

2.4 合并 PDF 列表接口

public byte[] **autoConcatPdfList**(String pdfListXml, String outputFilePath， String savedPdfId , String operatorCode, String channelCode) throws Exception
使用场景：主要用于合并多个 pdf 文件
入参：
pdfListXml：<List>< Pdf >pdf1Base64</ Pdf >< Pdf >pdf2Base64</ Pdf ></List>
如果 pdf 的字节长度大于 18 小于 32 时，按文件 pdfId 处理；
需要从 TEMP_FILE_PDF 中获取 pdf 数据
如果 pdf 的字节长度大于 40 小于 500 时，按照如下格式处理：
1：文件路径“/home/1.pdf”或者“D:/1.pdf”
outputFilePath：合成后的存放路径
savedPdfId：临时保存的 pdfId，可以为空
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空
正常返回：
合并后的 pdf
异常返回：
<Error>
<ErrorCode>errorCode</ErrorCode>
<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

2.5 PDF 自动化签章接口

public byte[] **sealAutoPdf**(byte[] pdf, String savedPdfId, String sealStrategyXml，String operatorCode, String channelCode) throws Exception
使用场景：自动化普通签章（印模图片可以外部传入）
入参：
pdf：pdf 文件数据或者是 pdfId（pdfId 表示之前保存的 savedPdfId）
savedPdfId：临时保存的 pdfId
sealStrategyXml：签章策略文件，见附件
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空

正常返回：
盖章后的 pdf 文件
异常返回：
<Error>
<ErrorCode>errorCode</ErrorCode>
<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

2.6 PDF 自动化复合签章接口

public byte[] **compoundSealAutoPdf**(String pdfBeanXml , String multiData，String compoundSealStrategyXml，String operatorCode, String channelBeanXml) throws Exception
使用场景：主要用于对 pdf 进行合成多媒体、一次性任意签章；如果有手写签名同一个 pdf 不能连续多次调用，否则 pdf 会报错
入参：
pdfBeanXml：格式如下：（下同）
<PdfBean>
<Pdf> pdf 单据 Base64 编码/或者 pdfId（之前保存的 savedPdfId）</Pdf>
<PdfOwnerPassword>pdf 拥有者密码 Base64 编码</PdfOwnerPassword>
<SavedPdfId>临时保存的 pdfId，可以为空</SavedPdfId>
<BizSerialNo>业务流水号，可以为空</BizSerialNo>
<BizTypeCode>业务类型编码(用来分类 pdf)，可以为空</BizTypeCode>
<CardNo>卡号，可以为空</CardNo>
<OutputFilePath>单据存放地址，可以为空</OutputFilePath>
<Reserved1>保留字段 1，可以为空</Reserved1>
<Reserved2>保留字段 2，可以为空</Reserved2>
<Reserved3>保留字段 3，可以为空</Reserved3>
</PdfBean>
Pdf：如果 pdf 的字节长度大于 18 小于 32 时，按文件 pdfId 处理（需要从临时 pdf 文件表中获取 pdf 数据）；如果 pdf 的字节长度大于 40 小于 500 时，按照文件路径格式处理（“/cfca/1.pdf” 或者“D:/cfca/1.pdf”或者“./cfca/1.pdf”）
OutputFilePath：可以是如下格式：
1.文件路径：/home/22.pdf
2.ftp 地址：ftp:/home/2015/201505/20150501/1.pdf 或者 sftp:/home…
3.无纸化系统自动创建，格式为 ftp:auto;auto，无纸化系统会根据 fileStoreMainDirectory||channelCode||bizTypeCode||operatorCode||yyyy||yyyyMM||yyyyMMdd 的任意组合创建目录

multiData: 需要合成到 pdf 的多媒体数据，格式同上，可以为空

compoundSealStrategyXml：复合证书签章策略文件，格式如下：
<List>证据证书签章策略文件、签章策略文件 </List>
operatorCode：
柜员编码或机构号，可多个，格式：operatorCode##operatorCode##
channelBeanXml：渠道信息，可以为空
<Channel>
<ChannelCode>渠道编码</ChannelCode>
<MerchantNo>商户号，可以为空</MerchantNo>
<MerchantName>商户名称，可以为空</MerchantName>
<TerminalNo>终端号，可以为空</TerminalNo>
</Channel>
正常返回：
盖章后的 pdf 文件
异常返回：
<Error>
< ErrorCode >errorCode</ ErrorCode >
<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

2.7 PDF 列表自动化复合签章接口

public String **compoundSealAutoPdfList**(String pdfBeanListXml, String multiData，String compoundSealStrategyXml，String operatorCode, String channelBeanXml) throws Exception
使用场景：一次性对多个 pdf 进行合成多媒体，任意签章时用到此接口；如果有手写签名同一个 pdf 不能连续多次调用，否则 pdf 会报错。
入参：
pdfBeanList：pdfBean 列表，格式（同上）
<List>
<PdfBean>
</PdfBean>
<PdfBean>
</PdfBean>
</List>
multiData: 需要合成到 pdf 的多媒体数据，格式同上，可以为空
compoundSealStrategyXml：复合证书签章策略文件，格式如下：
<List>证据证书 pdf 列表签章策略文件、pdf 列表签章策略文件 </List>
operatorCode：
操作员编码或机构号，可多个，格式为：operatorCode##operatorCode##...
channelBeanXml：渠道信息，格式同上，可以为空

正常返回：（Code 不等于 200 时为异常）

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
  <List>
    <PdfBean>
      <Pdf>pdf1Base64</ Pdf >
      <BizSerialNo >pdf1 存放到 ProofPdf 表中的 BizSerialNo </BizSerialNo >
    </PdfBean>
    <PdfBean>。。。</PdfBean>
  </List>
</Result>
```

注意：输入参数 OutputFilePath 有值时，PdfId 才会有值

抛出：

Exception

2.8 PDF 列表自动化签章分离式接口

public String **compoundSealAutoPdfListDetached**(String pdfBeanListXml, String multiData, String compoundSealStrategyXml , String operatorCode, String channelBeanXml) throws Exception

使用场景：一次性对多个 pdf 进行多媒体合成，任意签章时用到此接口；并且同一个 pdf 可以连续多次调用，多媒体数据会存放到数据库中。

入参：

pdfBeanList：pdfBean 列表，格式（同上）

```
<List>
  <PdfBean>
  </PdfBean >
  <PdfBean >
  </PdfBean >
</List>
```

multiData: 需要合成到 pdf 的多媒体数据，格式同上，默认放到数据库，可以为空

compoundSealStrategyXml：复合证书签章策略文件，格式如下：

<List>证据证书 pdf 列表签章策略文件、pdf 列表签章策略文件</List>

operatorCode：操作员编码或者机构号

channelBeanXml：渠道信息，可以为空，格式同上

```
  </Channel>
```

正常返回：（Code 不等于 200 时为异常）

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
```

```
  <List>
    <PdfBean>
        <Pdf>pdf1Base64</ Pdf >
        <BizSerialNo > pdf1 存放到 ProofPdf 表中的 BizSerialNo </BizSerialNo >
    </PdfBean >
    <PdfBean >
       < Pdf >pdf2Base64</BizSerialNo >
       < BizSerialNo >2</ BizSerialNo >
     </PdfBean>
  </List>
</Result>
```

抛出：

Exception

2.9 PDF 验章接口

public String **verifyPdfSeal**(byte[] pdf, String verifyStrategyXml， String operatorCode，String channelCode) throws Exception

入参：

pdf：pdf 文件或 pdfId 取字节

verifyStrategyXml：验章策略文件，见附件

operatorCode：操作员编码或者机构号

channelCode：渠道编码，可以为空

返回：（Code 不等于 200 时为异常）

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
  <CertDN>dn1 |+| dn2</CertDN>
</Result>
```

抛出：

Exception

2.10 撤销 PDF 印章接口

public byte[] **revokePdfSeal**(byte[] pdf, int count, String operatorCode) throws Exception

入参：

pdf：pdf 文件流

count：需要撤销印章的个数，0：全部撤销，默认：1

operatorCode：操作员编码或者机构号

正常返回：

Pdf 文件
异常返回：
<Error>
<ErrorCode>errorCode</ ErrorCode >
<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

3 无纸化扩展接口

3.1 自动化计算 PDF 签章 HASH 值接口

public String **autoCalculatePdfHash**(byte[] pdf, byte[] x509Cert, String strategyXml, String operatorCode, String channelCode) throws Exception
使用场景：主要用于想使用客户端签名控件签名，后端合成印章的场景，此接口优点在于最终客户的印章不用托管到平台，可以直接使用最终客户的 key 来做签名。此接口需要和下一个接口配合使用。
入参：
pdf：pdf 文件
x509Cert：公钥证书，rsa 算法时可以为空，为空时外部需要传入 Pkcs7 签名，sm2 算法时不能为空，因为计算 hash 需要公钥证书。
strategyXML：策略文件，格式如下：

```
<Request>
  <SealReason>签章原因</SealReason>
  <SealLocation>签章地点</SealLocation>
  <SealImage>印模 Base64 编码</SealImage>
  <SealLayer2Text>印章显示文本，此文本会替代原有图片<SealLayer2Text>
  <SealLayer2TextStyle>font-size:10;width:100<SealLayer2TextStyle>
  <Type>2=坐标签章, 3=关键字签章</Type>
  <Page></Page>//pdf 页码
  <LX></LX>//X 坐标
  <LY></LY>//Y 坐标
  <Keyword></Keyword>//只能支持文档中有一个关键字
  <LocationStyle></LocationStyle>//风格，位置风格,上:U,下:D,左:L,右:R,中:C
  <OffsetX></OffsetX>//X 轴偏移量
  <OffsetY></OffsetY>//Y 轴偏移量
```

<HashAlg>sha1/sha256/sm3 </HashAlg>//hash 算法
</Request>
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空
正常返回：（Code 不等于 200 时为异常）
<Result>
<Code>200</Code>
<Message>successfully!</Message>
<PdfSealHash>hash 值 Base64 编码字符串</ PdfSealHash>
<PdfSealSource>原文 Base64 编码</ PdfSealHash>
<PdfHash>pdf 原文的 hashBase64</PdfHash>
<Id>保留值标识</ Id>
</Result>
抛出：
Exception

3.2 PDF 合成外部签名接口

public byte[] **synthesizeOuterSignature**(byte[] signature, String id, String operatorCode, String channelCode) throws Exception
入参：
signature：p1 签名值或者 p7 签名值，计算 hash 时如果不带证书，此处必须是 p7 格式
id：保留值标识
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空
正常返回：
Pdf 文件
异常返回：
<Error>
<ErrorCode>errorCode</ ErrorCode >
<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

3.3 PDF 合成外部签名并签章接口

public byte[] **synthesizeOuterSignatureAndSealPdf**(byte[] signature, String pdfBeanXml, String sealStrategyListXml, String operatorCode, String channelCode) throws Exception

入参：
signature：p1 签名值或者 p7 签名值，计算 hash 时如果不带证书，此处必须是 p7 格式
pdfBeanXml：格式如下：
<PdfBean>
<PdfId>保留值标识</PdfId>
<SavedPdfId>临时保存的 pdfId，可以为空</SavedPdfId>
<BizSerialNo>业务流水号，可以为空</BizSerialNo>
<BizTypeCode>业务类型编码(用来分类 pdf)，可以为空</BizTypeCode>
<OutputFilePath>单据存放地址，可以为空</OutputFilePath>
</PdfBean>
sealStrategyListXml：签章策略列表，可以为空，格式如下：
<List>签章策略</List>
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空
正常返回：
Pdf 文件
异常返回：
<Error>
<ErrorCode>errorCode</ ErrorCode >
<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

3.4 PDF 自动化复合计算 Hash 接口

public String **compoundAutoCalculatePdfHash**(byte[] pdf, byte[] x509Cert, String multiData，String proofSealStrategyXml，String operatorCode, String channelCode) throws Exception
使用场景：主要用于对 pdf 进行复合计算 hash，多媒体和证据 hash 会被放到 pdf 附件中
入参：
pdf：pdf 文件
x509Cert：公钥证书，可以为空，为空时外部需要传入 Pkcs7 签名
multiData：需要合成到 pdf 的多媒体数据，格式同上，可以为空
proofSealStrategyXml：证据证书签章策略文件
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空
正常返回：
盖章后的 pdf 文件

异常返回：

<Result>
<Code>200</Code>
<Message>successfully!</Message>
<PdfSealHash>hash 值 Base64 编码字符串</ PdfSealHash>
<PdfSealSource>原文 Base64 编码</ PdfSealHash>
<PdfHash>pdf 原文的 hashBase64</PdfHash>
<Id>保留值标识</ Id>
</Result>

抛出：

Exception

3.5 PDF 自动化 Hash 签名接口(带证据 hash)

public String **signAutoPdfProofHash**(String pdfHashListXml,String userInfoXml, String proofHashXml, String authCodeXml, String operatorCode, String channelCode) throws Exception

使用场景：主要用于根据场景申请证书然后对 Hash 进行签名。

入参：

pdfHashXml：格式如下：

<List>
<PdfHash>
<PdfSealHash>待签名 hash 的 base64</PdfSealHash>
<PdfSealSource>待签名原文的 base64</ PdfSealSource>
<BizTypeCode>业务类型编码，可以为空</ BizTypeCode>
</PdfHash>
</List>

userInfoXml：

<UserInfo>
<UserName>用户名称</UserName>
<IdentificationType>证件类型</IdentificationType>
<IdentificationNo>证件号码</ IdentificationNo>
<KeyAlg>算法</KeyAlg>
<InvokeWay>调用方式，1：local；2：remote </InvokeWay>
</UserInfo>

proofHashXml：

< ProofHashXml>

〈Proof fileName=“handwritingData.dat" hash="ffffffffffxxxxxxxxx” descr=“aa”〉</Proof〉

<Proof fileName =" handwritingImage.png" hash="ggggggaaaaaaaaa" descr=“aa”></Proof>

</ProofHashXml>

authCodeXml，格式如下：

<AuthCode>

<AuthCodeValue>验证码(6 位)</AuthCodeValue>
< AuthCodeId>验证码唯一识别 ID</AuthCodeId>
</AuthCode>
operatorCode：
柜员编码或机构号，可多个，格式：operatorCode##operatorCode##
channelCode：
正常返回：

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
  <List>
    <SignedBean>
        <Pkcs7>场景证书 p7 签名</Pkcs7>
        <Pkcs1>场景证书 p1 签名</Pkcs1>
     </SignedBean>
  </List>
</Result>
```

抛出：
Exception

3.6 PDF 自动化 Hash 签名接口

public String **signAutoPdfHash**(String pdfHashSignListXml, String authCodeXml, String operatorCode, String channelCode) throws Exception
使用场景：主要用于对 pdf 的 hash 进行签名
入参：
pdfHashSignListXml：格式如下：

```
<List>
    <PdfHashSign>
        <PdfSealHash>待签名 hash 的 base64</PdfSealHash>
        <PdfSealSource>待签名原文的 base64</ PdfSealSource>
        <BizTypeCode>业务类型编码，可以为空</ BizTypeCode>
        <SealCode>印章编码</SealCode>
        <SealPassword>印章密码</SealPassword>
        <InvokeWay>调用方式，1：local；2：remote </InvokeWay>
    </PdfHashSign>
</List>
```

authCodeXml，格式如下：

```
    <AuthCode>
        <AuthCodeValue>验证码(6 位)</AuthCodeValue>
        < AuthCodeId>验证码唯一识别 ID</AuthCodeId>
    </AuthCode>
```

operatorCode：
柜员编码或机构号，可多个，格式：operatorCode##operatorCode##
channelCode：
正常返回：

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
  <List>
    <SignedBean>
        <Pkcs7>场景证书 p7 签名</Pkcs7>
        <Pkcs1>场景证书 p1 签名</Pkcs1>
    </SignedBean>
  </List>
</Result>
```

抛出：
Exception

3.7 自动化发送短信接口

public String **sendShortMessage**(String sendMessageXml, String operatorCode, String channelCode) throws Exception
使用场景：发送 6 位短信验证码
入参：
sendMessageXml：格式如下：

```
  <Request>
      <PhoneNo>电话号码</PhoneNo>
      <SealCode>印章编码</SealCode>
      <UserName>用户名称</UserName>
      <IdentificationType>证件类型</IdentificationType>
      <IdentificationNo>证件号码</ IdentificationNo>
      <SealFlag>印章标识</SealFlag>
      <ValidityPeriod>有效期，范围（30 秒-180 秒），默认 60 秒</ValidityPeriod>
      <TemplateCode>短信模板编码</TemplateCode>
  </Request>
```

operatorCode：
柜员编码或机构号，可多个，格式：operatorCode##operatorCode##
channelCode：
正常返回：

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
```

<AuthCodeId>验证码 id</AuthCodeId>
<ValidityPeriod>有效期（秒）</ValidityPeriod>
</Result>
抛出：
Exception

4 无纸化辅助接口

4.1 HTML/WORD 转换 PDF 功能

public byte[] **transformToPdf**(byte[] source，String fileType, String savedPdfId, String operatorCode, String channelCode) throws Exception
入参：
source：待转换的内容的字节 UTF-8 编码
fileType：文件类型，可以取值：html：java 程序转换， html2：调用 OpenOffice 服务转换， word2：调用 OpenOffice 服务转换。
savedPdfId：临时保存的 pdfId，可以为空
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空
正常返回：
转换后的 pdf 文件
异常返回：
<Error>
<ErrorCode>errorCode</ErrorCode>
<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

4.2 IMAGE 转换 PDF 功能

public byte[] **transformImageToPdf**(String imageListXml, String savedPdfId, String operatorCode, String channelCode) throws Exception
入参：
imageListXml：图片 Base64 列表格式如下：
<List><Image>imageBase64</Image>…..</List>
savedPdfId：临时保存的 pdfId，可以为空
operatorCode：操作员编码或者机构号

channelCode：渠道编码，可以为空
正常返回：
转换后的 pdf 文件
异常返回：
<Error>
<ErrorCode>errorCode</ErrorCode>
<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

4.3 PDF 自动化转图片接口

public String **transformPdfToImage**(byte[] pdf, String pageNo, String operatorCode, String channelCode) throws Exception
入参：
pdf：pdf 文件流
pageNo：需要转图片的页码，值为-1 或者 0 时代表全部
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空
正常返回：（Code 不等于 200 时为异常）
<Result>
<Code>200</Code>
<Message>successfully!</Message>
<Image>imageBase64(图片的 base64 编码)</ Image >
</Result>
抛出：
Exception

4.4 PDF 自动化骑缝章签章接口

public byte[] **sealAutoCrossPdf**(byte[] pdf, String savedPdfId, String crossSealStrategyXml, String operatorCode, String channelCode) throws Exception
入参：
pdf：pdf 文件流
savedPdfId：临时保存的 pdfId，可以为空
crossSealStrategyXml：骑缝签章策略文件，见附件
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空
正常返回：

盖章后的 pdf 文件
异常返回：
<Error>
<ErrorCode>errorCode</ErrorCode>
<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

4.5 PDF 添加水印接口

public byte[] **addWaterMarkToPdf**(byte[] pdf, String savedPdfId, String waterMarkStrategyXml, String operatorCode, String channelCode) throws Exception
入参：
pdf：pdf 文件流
savedPdfId：临时保存的 pdfId，可以为空
waterMarkStrategyXml：添加水印策略文件，见附件
operatorCode：操作员编码或者机构号
channelCode：渠道编码，可以为空
正常返回：
添加水印后的 pdf 文件
异常返回：
<Error>
<ErrorCode>errorCode</ErrorCode>
<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

4.6 获取时间戳 URL 接口

public String **getTimestampURL**(String flag, String operatorCode) throws Exception
入参：
flag：时间戳标识
operatorCode：操作员编码
channelCode：渠道编码，可以为空
返回：（Code 不等于 200 时为异常）
<Result>

```
  <Code>200</Code>
  <Message>successfully!</Message>
  <Timestamp>
    <TimestampURL></TimestampURL>
    <TimestampUser></TimestampUser>
    <TimestampPassword></TimestampPassword>
  </Timestamp>
</Result>
```

抛出：

Exception

4.7 获取印章信息接口

public String **getSealInfo**(String sealCode，String operatorCode) throws Exception

入参：

sealCode：印章编码

operatorCode：操作员编码

正常返回：

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
  <SealInfoXml>
        <SealCode>印章编码</SealCode>
        <SealName>印章名称</SealName>
        <SealPassword>印章密码</SealPassword>
        <TellerCode>柜员编码</TellerCode>
        <OrgCode>机构编码</OrgCode>
        <OrgName>机构名称</OrgName>
        <Cert>公钥证书 Base64 编码</Cert>
        <Image>印模图片 Base64 编码</Image>
  </SealInfoXml>
</Result>
```

抛出：

Exception

4.8 获取印章信息列表接口

public String **getSealInfoList**(String requestXml，String operatorCode) throws Exception

入参：

requestXml：参数 Xml，可以为空，为空时查询所有印章，格式如下：

<Request>
<OrgCode>机构编码</OrgCode>//可以为空
<OrgName>机构名称</OrgName>//可以为空
< Request>
operatorCode：操作员编码
正常返回：

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
  <List>
    <SealInfoXml>
        <SealCode>印章编码</SealCode>
        <SealName>印章名称</SealName>
         <SealPassword>印章密码</SealPassword>
         <TellerCode>柜员编码</TellerCode>
         <OrgCode>机构编码</OrgCode>
         <OrgName>机构名称</OrgName>
         <Image>印模图片 Base64 编码</Image>
    </SealInfoXml>
  </List>
</Result>
```

抛出：
Exception

4.9 获取印章申请信息接口

public String **getSealApplyInfo**(String requestXml，String operatorCode) throws Exception
入参：
requestXml：参数 Xml，可以为空，为空时查询所有印章，格式如下：

<Request>
<SealCode>印章编码</SealCode>//可以为空
<UserName>机构名称</UserName>//印章编码为空时，如下不可以为空
<IdentificationType> 证件类型</IdentificationType>
<IdentificationNo> 证件号码 </IdentificationNo>
<SealFlage>印章标识</SealFlag>
< Request>
operatorCode：操作员编码
正常返回：

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
```

```
  <SealApplyInfoXml>
        <SealCode>印章编码 </SealCode>
        <SealPassword> 印章密码 </SealPassword>
        <UserName> 柜员编码 </UserName>
        <IdentificationType> 证件类型 </IdentificationType>
        <IdentificationNo> 证件号码 </IdentificationNo>
        <PhoneNo> 电话号码 </PhoneNo>
        <SealFlag> 印章标识 </SealFlag>
        <CertDn>证书 DN</CertDn>
        <CertSn>证书序列号</CertSn>
        <CertNotBefore></CertNotBefore>
        <CertNotAfter></CertNotAfter>
  </SealApplyInfoXml>
</Result>
```

抛出：

Exception

4.10 获取模板接口

public String **getTemplate**(String templateCode, String hashAlg, String operatorCode) throws Exception

入参：

templateCode：模板编码，多个模板号用逗号分隔，比如：tmp001,tmp002

hashAlg：hash 算法，支持 md5/sha1/sha256，可以为空

operatorCode：操作员编码

正常返回：（Code 不等于 200 时为异常）

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
  <Template>
    <TemplateCode></TemplateCode >模板编码
    <Version></Version>版本号
    <TemplateName></TemplateName>模板名称
    <TemplateData>TemplateDataBase64</TemplateData>模板内容
    <TemplateFormat></TemplateFormat>模板格式
    <HashValue>TemplateDataBase64 解 Base64 后的 hash Base64</HashValue>
  </Template>
  <Template>
    …
  </Template>
</Result>
```

抛出：

Exception

4.11 获取模板信息列表接口

public String **getTemplateInfoList**(String requestXml, String operatorCode) throws Exception
入参：
requestXml，格式如下：
<Request>
<TemplateFormat>格式，可以为空</TemplateFormat>
<HashAlg> hash 算法，支持 md5/sha1/sha256，可以为空</HashAlg>
</Request>
operatorCode：操作员编码
正常返回：（Code 不等于 200 时为异常）
<Result>
<Code>200</Code>
<Message>successfully!</Message>
<Template>
<TemplateCode></TemplateCode >模板编码
<Version></Version>版本号
<TemplateName></TemplateName>模板名称
<TemplateFormat></TemplateFormat>模板格式
<HashValue>TemplateDataBase64 解 Base64 后的 hash Base64</HashValue>
</Template>
<Template>
…
</Template>
</Result>
抛出：
Exception

4.12 自助生成图片接口

public byte[] **autoGenerateImage**(String imageStrategyXml, String operatorCode, String channelCode) throws Exception
入参：
imageStrategyXml：图片生成策略文件，见附件
operatorCode：操作员编码或机构号
channelCode：渠道编码，可以为空
正常返回：

图片
异常返回：
<Error>
<ErrorCode>errorCode</ ErrorCode >
<ErrorMessage>errorMessage</ErrorMessage>
</Error>
抛出：
Exception

4.13 申请并下载证书接口

public String **applyAndDownloadCert**(String applyCertStrategyXml，String operatorCode, String channelCode) throws Exception
入参：
applyCertStrategyXml：证书申请策略文件，见附件
operatorCode：操作员编码
channelCode：渠道编码，可以为空
返回：（Code 不等于 200 时为异常）
<Result>
<Code>200</Code>
<Message>successfully!</Message>
<CertType></CertType>
<Cert>Base64</ Cert>
<EncryptionCert> Base64</EncryptionCert>
<EncryptionPrivateKey> Base64</EncryptionPrivateKey>
</Result>
抛出：
Exception

4.14 获取关键字在 PDF 中的坐标

public String **getKeywordLocationsInPdf**(byte[] pdf, String keyword, String operatorCode) throws Exception
入参：
pdf：pdf 文件流
keyword：关键字
operatorCode：操作员编码
正常返回：（Code 不等于 200 时为异常）
<Result>
<Code>200</Code>
<Message>successfully!</Message>

```
  <List>
    <Location>
        <PageNo> 页码
        <LX>   横坐标（以左下为参考）
        <LY>  纵坐标（以左下为参考）
        <Width>页面宽度
        <Height>页面高度
    </Location>
  </List>
</Result>
```

抛出：

Exception

4.15 获取广告

public String **getAdvertisement**(String requestXml, String operatorCode) throws Exception

入参：

requestXml，请求 xml，格式如下：

```
  <Request>
      <Code>广告编码，可以多个，以逗号分隔</Code>
      <Version>版本号，可以为空</Version>
      <ForceReturn>true/false</ForceReturn>
      //true：version 不为空时强制返回对应版本广告，否则抛异常，version 为空返回最新版本广告
      //false：比对 version 和库中最新版本，如果相同不返回广告，否则返回最新版本
  </Request>
```

正常返回：

广告文件

正常返回：（Code 不等于 200 时为异常）

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
  <Advertisement>
    <Code></Code>
    <Name>名称</Name>
    <FileFormat></FileFormat>
    </FileData>dataBase64</FileData>
    <Type>广告类型</Type>
    <Version>版本号</Version>
    <Status>状态</Status>
```

```
  </Advertisement>
  <Advertisement>
  ……
  </Advertisement>
</Result>
```

抛出：

Exception

4.16 获取广告列表

public String **getAdvertisementInfoList**(String requestXml, String operatorCode) throws Exception

入参：

requestXml，请求 xml，格式如下：

```
<Request>
      <HashAlg > hash 算法，支持 md5/sha1/sha256，可以为空</HashAlg>
</Request>
```

正常返回：

广告文件

正常返回：（Code 不等于 200 时为异常）

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
  <Advertisement>
    <Code>编码</Code>
    <Name>名称</Name>
    <FileFormat></FileFormat>
    <Type>广告类型</Type>
    <Version>版本号</Version>
    <Status>状态</Status>
    <HashValue>TemplateDataBase64 解 Base64 后的 hash Base64</HashValue>
  </Advertisement>
  <Advertisement>
  ……
  </Advertisement>
</Result>
```

抛出：

Exception

5 无纸化查询接口

5.1 查询单据信息列表接口

public String **queryProofInfoList**(String queryXml，String operatorCode) throws Exception
使用场景：用来查询单据信息列表
入参：
queryXml：格式如下：

```
<QueryXml>
  <StartTime>yyyyMMddHHmmss</StartTime>
  <EndTime> yyyyMMddHHmmss </EndTime>
  <BizTypeCode>业务类型编码</BizTypeCode>
  <ChannelCode>渠道编码</ChannelCode>
  <PageNo>页码</PageNo>//默认每页 100 条
</QueryXml>
```

operatorCode：操作员编码或者机构号
返回：（Code 不等于 200 时为异常）

```
<Result>
  <Code>200</Code>
  <Message>successfully!</Message>
  <TotalPage></TotalPage>//总页数
  <TotalCount></TotalCount>//总条数
  <List>
    <ProofInfo>
      <BizSerialNo></BizSerialNo>
      <BizTypeCode>业务类型编码</BizTypeCode>
      <Status>状态</Status>
      <CreateTime></CreateTime>
      <MerchantNo></MerchantNo>
      <TerminalNo></MerchantNo>
    </ProofInfo>
  </List>
</Result>
```

抛出：
Exception

5.2 查询单据接口

http://ip:port/PaperlessServer/PaperlessQueryServlet?bizSerialNo=xxx

6 附件

签章策略文件：

<Request>		
	<Type>	签章类型(不能为空),1=空白标签签章,2=坐标签章,3=关键字签章
	<SealCode>	印章编码（不能为空）
	<SealPassword>	印章密码（不能为空）
	<SealImage>	印章图片的 base64 编码，可以为空，此图片会替代原有印模
	<SealLayer2Text>	印章显示文本，此文本会替代原有图片
	<SealLayer2TextStyle>	font-size:10;width:100
	<SealPerson>	签章人
	<SealLocation>	签章地点
	<SealReason>	签章理由
	<FillOpacity>	透明度 0-1.0f
	<Visible>	是否显示，true or false
	<SealImageSize>	印章的大小，比如：100，代表 100px。
	<BusinessCode>	业务码：可以是验证码、查询码
	<BusinessCodeStyle>	font-size:10;x-ratio:0.5;y-ratio:0.5;
	<Page>	页数，按坐标签章时不能为空
	<LX>	左侧的 x 坐标
	<LY>	左侧的 y 坐标
	<Keyword>	关键字，按关键字签章时不能为空
	<LocationStyle>	位置风格：上:U；下:D；左:L；右:R；中:C；默认：C
	<OffsetX>	横轴偏移，默认为 0

	<OffsetY>	纵轴偏移，默认为 0

Pdf 位置列表签章策略文件：

<Request>		
	<SealCode>	印章编码（不能为空）
	<SealPassword>	印章密码（不能为空）
	<SealImage>	印章图片的 base64 编码，可以为空，此图片会替代原有印模
	<SealLayer2Text>	印章显示文本，此文本会替代原有图片
	<SealLayer2TextStyle>	font-size:10;width:100
	<SealPerson>	签章人
	<SealLocation>	签章地点
	<SealReason>	签章理由
	<FillOpacity>	透明度 0-1.0f
	<Visible>	是否显示，true or false
	<SealImageSize>	印章的大小，比如：100，代表 100px。
	<BusinessCode>	业务码：可以是验证码、查询码
	<BusinessCodeStyle>	font-size:10;x-ratio:0.5;y-ratio:0.5;
	<Location>	
	<Type>	2=坐标签章,3=关键字签章
	<PdfIndex>	取值 1、2、3、4 等，-1：代表全部
	<Page>	页数，按坐标签章时不能为空
	<LX>	左侧的 x 坐标
	<LY>	左侧的 y 坐标
	</Location>	
	<Location>	
	<Type>	2=坐标签章,3=关键字签章
	<PdfIndex>	取值 1、2、3、4 等，-1：代表全部

	<Keyword>	关键字，按关键字签章时不能为空
	<LocationStyle>	位置风格：上:U；下:D；左:L；右:R；中:C；默认：C
	<OffsetX>	横轴偏移，默认为 0
	<OffsetY>	纵轴偏移，默认为 0
	</Location>	

证据证书签章策略文件：

<Request>		
	<Type>	类型，1=空白标签签章,2=坐标签章,3=关键字签章
	<HandwritingImage>	手写签名图片 Base64
	<SealPerson>	签章人
	<SealLocation>	签章地点
	<SealReason>	签章理由
	<IdentificationType>	证件类型
	<IdentificationNo>	证件号码
	<PhoneNo>	电话号码
	<KeyAlg>	RSA/SM2
	<FillOpacity>	透明度 0-1.0f
	<Visible>	是否显示,true or false
	<Page>	页数,按坐标签章时不能为空
	<LX>	左侧的 x 坐标
	<LY>	左侧的 y 坐标
	<Keyword>	关键字,按关键字签章时不能为空
	<LocationStyle>	位置风格：上:U；下:D；左:L；右:R；中:C；默认：

		C
	<OffsetX>	横轴偏移，默认为 0
	<OffsetY>	纵轴偏移，默认为 0
	< ProofHashXml> <Proof fileName="handwritingData.dat" hash="ffffffffffxxxxxxxxx" descr="aa"></Proof> <Proof fileName =" handwritingImage.png" hash="ggggggggggaaaaaaaaa" descr="aa"> </ProofHashXml>	证据 hash，会保存到 pdf 的附件，存放到证书的扩展域 2.16.156.112554.9.1 中

证据证书 pdf 列表签章策略文件：

<Request>		
	<HandwritingImage>	手写签名图片 Base64
	<SealPerson>	签章人
	<SealLocation>	签章地点
	<SealReason>	签章理由
	<IdentificationType>	证件类型
	<IdentificationNo>	证件号码
	<PhoneNo>	电话号码
	<KeyAlg>	RSA/SM2
	<FillOpacity>	透明度 0-1.0f
	<Visible>	是否显示，true or false
	<Location>	
	<Type>	2=坐标签章，3=关键字签章
	<PdfIndex>	取值 1、2、3、4 等，-1：代表全部
	<Page>	页数，按坐标签章时不能为空
	<LX>	左侧的 x 坐标
	<LY>	左侧的 y 坐标
	</Location>	
	<Location>	

	<Type>	2=坐标签章,3=关键字签章
	<PdfIndex>	取值 1、2、3、4 等，-1：代表全部
	<Keyword>	关键字，按关键字签章时不能为空
	<LocationStyle>	位置风格：上:U；下:D；左:L；右:R；中:C；默认：C
	<OffsetX>	横轴偏移，默认为 0
	<OffsetY>	纵轴偏移，默认为 0
	</Location>	
	< ProofHashXml> <Proof fileName="handwritingData.dat" hash="ffffffffffxxxxxxxxx" descr="aa"></Proof> <Proof fileName =" handwritingImage.png" hash="ggggggaaaaaaaaaa" descr="aa"></Proof> </ProofHashXml>	证据 hash，会保存到 pdf 的附件，存放到证书的扩展域 2.16.156.112554.9.1 中

验章策略文件：

<Request>		
	<SealVerifyType	0-验章的签名， 1- 验证签章时间点证书是否有效 2- 验证证据证书中的证据和pdf中的证据是否一样，验证pdf内容Hash和证书中的Hash 3-验证书链， 4-验CRL
</Request>		

证书申请策略文件：

<Request>		
	<CustomerType>	客户类型，1：个人普通；2：企业普通 7：个人场景，8：企业场景
	<CertLevel>	证书类型，1：普通；2：高级
	<KeyAlg>	密钥算法：RSA，SM2（只能是高级）
	<KeyLength>	密钥长度：1024，2048，256
	<UserName>	用户名

	<IdentificationType>	证件类型
	<IdentificationNo>	证件号码
	<PhoneNo>	电话号码
	<SelfExtValue>	自定义扩展域值
	<P10>	申请书 Base64
	<InvokeWay>	调用方式，1：local；2：remote

骑缝章签章策略文件：

<Request>		
	<SealCode>	印章编码（不能为空）
	<SealPassword>	印章密码（不能为空）
	<SealPerson>	签章人
	<SealLocation>	签章地点
	<SealReason>	签章理由
	<FromPage>	起始页，如果为 FromPage 和 ToPage 都为 0 时代表全部页面都盖章
	<ToPage>	结束页
	<CrossStyle>	骑缝章风格 1：上下各一半 2：左右各一半 3：上下骑缝 4：左右骑缝 5：左骑缝 6：右骑缝

添加水印策略文件：

<Request>		
	<FromPage>	起始页,0 代表全部
	<ToPage>	结束页，0 代表全部
	<AbsoluteX>	水印 X 坐标
	<AbsoluteY>	水印 Y 坐标
	<ApartX>	X 坐标相距（用于控制水印之间 X 距离）
	<ApartY>	Y 坐标相距（用于控制水印之间 Y 距离）
	<Alpha>	透明度，取值 0-1
	<FitWidth>	宽度
	<FitHeight>	高度

	<RotationDegree>	旋转度
	<WaterMarkImage>	水印图片 Base64 编码
	<WaterMarkText>	水印文本

图片生成策略文件：

<Request>		
	<ImageName>	长方形换行字符“##”
	<ImageName2>	
	<ImageWidth>	图片宽
	<ImageHeight>	图片高
	<ImageShape>	1：方型 11：方形（带框） 2：长方形（不带“章”字） 21：长方形（带框） 3：圆形（不带五角星） 4：圆形（带五角星）
	<Color>	十六进制，默认FF0000（适用于方章和长方形章）
	<FontSize>	字体大小，默认100（适用于方章和长方形章）
<Request>		