**北京江南天安科技有限公司**

**应急广播签名验证接口**

**调用说明**



北京江南天安科技有限公司

**2017-05-31**

目录

[1 概要说明 4](#_Toc22270)

[1.1 范围 4](#_Toc10622)

[1.2 环境准备 4](#_Toc11671)

[1.2.1 运行环境 4](#_Toc32495)

[1.2.2 调用准备 4](#_Toc16556)

[2 接口应用说明 5](#_Toc4632)

[2.1 接口文件说明 5](#_Toc21687)

[2.2 配置文件说明 5](#_Toc9907)

[2.2.1 sjj1313配置文件说明 5](#_Toc29225)

[2.2.2 sjj1507配置文件说明 5](#_Toc24246)

[2.3 接口部署说明 6](#_Toc12978)

[2.3.1 C语言版本动态库调用部署 6](#_Toc5765)

[2.3.2 Java版本接口调用部署 6](#_Toc1449)

[2.4 接口调用流程说明 7](#_Toc31377)

[2.4.1 C语言版本接口调用流程 7](#_Toc24143)

[2.4.2 JAVA版本接口调用流程 7](#_Toc3943)

[3 C语言功能接口说明 1](#_Toc14126)

[3.1 打开密码设备 1](#_Toc16079)

[3.2 关闭设备 1](#_Toc2081)

[3.3 计算数据签名（过时） 1](#_Toc27510)

[3.4 验证签名值（过时） 2](#_Toc9161)

[3.5 计算数据签名（字符串形式）（过时） 2](#_Toc13133)

[3.6 验证签名值（字符串形式）（过时） 3](#_Toc31994)

[3.7 计算平台数据签名 3](#_Toc4662)

[3.8 平台验证数据签名 4](#_Toc10988)

[3.9 十六进制数据转换为字符串（BCD -> HEX） 4](#_Toc2556)

[3.10 字符串转换为十六进制数据（HEX -> BCD） 5](#_Toc6627)

[3.11 对数据进行BASE64编码 5](#_Toc10452)

[3.12 对数据进行BASE64解码 6](#_Toc11840)

[3.13 计算数据摘要 6](#_Toc31921)

[4 JAVA版功能接口说明 7](#_Toc21747)

[4.1 打开密码设备 7](#_Toc8802)

[4.1.1 连接SJJ1313密码器设备（过时） 7](#_Toc5284)

[4.1.2 带参数的设备连接方法 7](#_Toc22177)

[4.2 关闭密码设备 7](#_Toc3347)

[4.3 计算数据签名（过时） 7](#_Toc24685)

[4.4 验证数据签名（过时） 8](#_Toc32380)

[4.5 计算数据签名（字符串形式）（过时） 9](#_Toc10091)

[4.6 验证数据签名（字符串形式）（过时） 9](#_Toc251)

[4.7 计算平台数据签名 10](#_Toc7717)

[4.8 平台验证数据签名 10](#_Toc5195)

[4.9 十六进制字符串数据编码（BCD -> HEX） 10](#_Toc10073)

[4.10 十六进制字符串数据解码（HEX -> BCD） 11](#_Toc12893)

[4.11 Base64编码 11](#_Toc5821)

[4.12 Base64解码 11](#_Toc9093)

[4.13 计算数据摘要 11](#_Toc12514)

[5 附录 1](#_Toc2399)

[5.1 SJJ1313错误代码说明 1](#_Toc27413)

[5.2 SJJ1507错误代码说明 2](#_Toc32158)

[5.3 应急广播命令错误码说明 4](#_Toc29556)

# 概要说明

## 范围

此文档用于描述不同密码设备（包括SJJ1313密码器、SJJ1507密码机等）为应急广播业务定制的签名验证功能的功能接口的定义及使用说明。除此以外的其他场景不适用于此文档。

## 环境准备

### 运行环境

本接口适用平台包括Windows及Linux平台的32和64位环境，除此之外的其他系统平台不在支持范围之内。

### 调用准备

调用本接口之前需根据不同型号的安装部署说明对设备进行初始化、安装、启动及配置操作，确保设备罐装正确的设备证书及信任证书。

# 接口应用说明

## 接口文件说明

表2-1 SJJ1313和SJJ1507C语言动态库文件列表

|  |  |  |  |
| --- | --- | --- | --- |
| **设备类型** | **操作系统** | **文件名** | **描述** |
| SJJ1313 | win32/win64 | dllSDF.dll | 设备基础功能动态库 |
| libTassYJGBCmd\_SJJ1313.h | 应急广播功能接口头文件 |
| pci\_api.dll | 底层依赖动态库 |
| libTassYJGBCmd\_SJJ1313.dll | 应急广播功能接口动态库 |
| libTassYJGBCmd\_SJJ1313.lib | 应急广播功能接口链接库 |
| linux32/linux64 | libTassSDF.so | 设备基础功能动态库 |
| libTassYJGBCmd\_SJJ1313.h | 应急广播功能接口头文件 |
| libTassYJGBCmd\_SJJ1313.so | 应急广播功能接口动态库 |
| SJJ1507 | win32/win64 | libTassSDF.dll | 设备基础功能接口动态库 |
| libTassYJGBCmd\_SJJ1507.h | 应急广播功能接口头文件 |
| libTassYJGBCmd\_SJJ1507.dll | 应急广播功能接口动态库文件 |
| libTassYJGBCmd\_SJJ1507.lib | 应急广播功能接口链接库 |
| linux32/linux64 | libTassSDF.so | 设备基础功能动态库 |
| libTassYJGBCmd\_SJJ1507.h | 应急广播功能接口头文件 |
| libTassYJGBCmd\_SJJ1507.so | 应急广播功能接口动态库文件 |

**表2-2 Java接口库文件列表**

|  |  |
| --- | --- |
| **文件名** | **描述** |
| jna-4.2.2.jar | JNA接口库 |
| YingJGBCALLDLL20170605.jar | Java版功能接口库 |

## 配置文件说明

### sjj1313配置文件说明

sjj1313配置文件为cardcfg.ini，所在的位置为sjj1313文件夹的各个子目录下，配置文件内容如下：

host=127.0.0.1

port=8888

其中host为设备的服务IP地址，port为服务端口号，在使用时候请根据实际情况修改配置文件中的host以及port。

### 2.2.2 sjj1507配置文件说明

Sjj1507配置文件为tacipher.ini，所在的位置为sjj1507文件夹的各个子目录下，配置文件内容如下：

timeout=60

ciphers=1

worktype=2

sockets=20

msgheadlen=0

encode=1

long\_con=0

level=1

host1=124.127.49.181

port1=8118

其中ciphers为可连接的密码设备的数量（本接口支持的数量为1），host1为设备的服务IP地址，port1为服务端口号，在使用时候请根据实际情况修改配置文件中的host1和port1。

## 接口部署说明

### C语言版本动态库调用部署

调用C语言动态库根据运行环境及应用系统架构设计选择对应平台的动态库版本并将对应的动态库文件置于相应位置进行调用即可。

需要注意的是，**各动态库所需的配置文件需与动态库文件处于相同目录下**。

### Java版本接口调用部署

Java版本接口为采用JNA对C语言版本的动态库进行封装，因此依赖内容包括：

* C语言动态库
  + Windows 环境下 32 位 JDK 调用时，将所需型号设备的DLL动态库文件及配置文件拷贝至“C:\Windows\System32”目录下；
  + Windows环境的 64 位 JDK 调用时，将所需型号设备的DLL动态库文件及配置文件拷贝至“C:\Windows\SysWOW64”目录下；
  + Linux 环境下 32 位 JDK 调用时，将所需型号设备的SO动态库文件及配置文件拷贝至“/usr/lib”目录下；
  + Linux环境下 64 位 JDK 调用时，将所需型号设备的SO动态库文件及配置文件拷贝至“/usr/lib64”目录下。
* 接口配置文件 - **与动态库置于相同目录下**；
* JNA接口包 - 根据应用系统设计，放置于对应的依赖目录下；
* Java版本功能接口包 - 根据应用系统设计，放置于对应的依赖目录下。

## 接口调用流程说明

一般情况下，此接口的C语言版本与Java版本应分别遵循以下流程和注意事项进行调用。

### C语言版本接口调用流程

**调用流程：**

1. 应用系统初始化时，调用“打开密码设备”功能建立与密码设备的连接，并在多线程间共享获取到的设备句柄变量；
2. 根据需要创建工作线程；
3. 调用“计算平台数据签名”或“平台验证数据签名”功能；
4. 应用系统退出时调用“关闭设备”断开与密码设备的连接。

**注意事项：**

1. C语言版本接口调用时首先需确定自己所使用的设备型号，根据设备型号选择对应的动态库接口文件进行开发调用；
2. 本接口中的设备连接资源为进程内多线程共享，因此应用系统最好采用单进行的模式来调用本接口。

### JAVA版本接口调用流程

**调用流程：**

1. 应用系统初始化时，调用“打开密码设备”（带参数的设备连接方法）功能建立与密码设备的连接；
2. 根据需要创建工作线程；
3. 根据业务流程需要调用“计算平台数据签名”或“平台验证数据签名”功能；
4. 应用系统退出时，调用“关闭密码设备”功能断开与密码设备的连接。

**注意事项：**

Java接口对多个型号的设备进行统一封装，在调用“打开密码设备”功能时，需传入所调用的设备的类型标识来进行区分。详见功能接口详细说明。

# C语言功能接口说明

## 打开密码设备

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | int OpenDevice(void \*\*phDeviceHandle) | | |
| 描述： | 打开与密码设备间的连接 | | |
| 参数： | [out] | phDeviceHandle | 打开的设备句柄 |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： |  | | |

## 关闭设备

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | int CloseDevice(void \*hDeviceHandle); | | |
| 描述： | 关闭密码设备，释放相关资源 | | |
| 参数： | [in] | hDeviceHandle | 已打开的密码设备句柄 |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： |  | | |

## 计算数据签名（过时）

此版本中该方法标记为过时，建议调用函数 “Platform\_CalculateSignature”。

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | int GenerateSignatureWithDevicePrivateKey(  void \*hDeviceHandle,  int nDataType,  unsigned char \*pucData,  int nDataLength,  unsigned char \*pucCounter,  unsigned char \*pucSignCerSn,  unsigned char \*pucSignature); | | |
| 描述： | 使用设备证书对指令活消息数据进行签名运算 | | |
| 参数： | [in] | hDeviceHandle | 已打开的密码设备句柄 |
| [in] | nDataType | 待签名数据类型：  *0. 指令数据*  *1. 消息数据* |
| [in] | pucData | 待签名数据 |
| [in] | nDataLength | 待签名数据长度 |
| [out] | pucCounter | 签名计数器（4字节） |
| [out] | pucSignCerSn | 签名证书编号（6字节） |
| [out] | pucSignature | 签名结果数据（64字节） |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： |  | | |

## 验证签名值（过时）

此版本中该方法标记为过时，建议调用函数 “Platform\_VerifySignature”。

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | int VerifySignatureWithTrustedCert(  void \*hDeviceHandle,  int nDataType,  unsigned char \*pucData,  int nDataLength,  unsigned char \*pucCounter,  unsigned char \*pucSignCerSn,  unsigned char \*pucSignature); | | |
| 描述： | 使用密码设备中的可信证书验证数据签名 | | |
| 参数： | [in] | hDeviceHandle | 已打开的密码设备句柄 |
| [in] | nDataType | 待签名数据类型：  *0. 指令数据*  *1. 消息数据* |
| [in] | pucData | 待签名数据 |
| [in] | nDataLength | 待签名数据长度 |
| [in] | pucCounter | 签名计数器（4字节） |
| [in] | pucSignCerSn | 签名证书编号（6字节） |
| [in] | pucSignature | 签名值数据（64字节） |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： |  | | |

## 计算数据签名（字符串形式）（过时）

此版本中该方法标记为过时，建议调用函数 “Platform\_CalculateSignature”。

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | int GenerateSignatureWithDevicePrivateKey\_String(  void \*hDeviceHandle,  int nDataType,  char \*pcData,  char \*pcResult); | | |
| 描述： | 使用设备证书对指令活消息数据进行签名运算 | | |
| 参数： | [in] | hDeviceHandle | 已打开的密码设备句柄 |
| [in] | nDataType | 待签名数据类型：  *0. 指令数据*  *1. 消息数据* |
| [in] | pcData | 待签名数据 |
| [out] | pcResult | 签名值数据，格式:[N字节签名数据原文 || 4字节记数器 || 6字节签名证书号 || 64字节签名值] |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： |  | | |

## 验证签名值（字符串形式）（过时）

此版本中该方法标记为过时，建议调用函数 “Platform\_VerifySignature”。

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | int VerifySignatureWithTrustedCert\_String(  void \*hDeviceHandle,  int nDataType,  char \*pcData,  char \*pcSignature); | | |
| 描述： | 使用设备中的可信证书验证签名 | | |
| 参数： | [in] | hDeviceHandle | 已打开的密码设备句柄 |
| [in] | nDataType | 待签名数据类型：  *0. 指令数据*  *1. 消息数据* |
| [in] | pcData | 待签名数据 |
| [in] | pcSignature | 签名值数据，格式:[N字节签名数据原文 || 4字节记数器 || 6字节签名证书号 || 64字节签名值] |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： |  | | |

## 计算平台数据签名

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | int [Platform](http://www.baidu.com/link?url=7AqMrcWdJpnwkHbBmEIsyUn4FOKfJOQmzOmajXbWPffr61Cl9CMeK1m7N2bugeMPZB40QMkLx6NxObNwZ2-vPDZ87UecMVVNLMGfLSDBv6iEnBiN_CnQrPeGFaTOXSs_)\_CalculateSignature(  void \*hDeviceHandle,  int nDataType,  unsigned char \*pucData,  int nDataLength,  char \*pcSignature,  int \*pnSignatureLength); | | |
| 描述： | 使用设备中的可信证书验证签名 | | |
| 参数： | [in] | hDeviceHandle | 已打开的密码设备句柄 |
| [in] | nDataType | 待签名数据类型：  *0. 指令数据*  *1. 消息数据* |
| [in] | pucData | 签名数据原文 |
| [in] | nDataLength | 输入数据字节长度 |
| [out] | pcSignature | Base64编码的签名值数据（带字符串结束符） |
| [i/o] | pnSignatureLength | 输入：签名值缓冲区指针长度  输出：签名值实际字符串长度 |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： | 输出签名结果格式为 Base64( 4B Counter || 6B CertID || 64B Signature ) | | |

## 平台验证数据签名

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | int [Platform](http://www.baidu.com/link?url=7AqMrcWdJpnwkHbBmEIsyUn4FOKfJOQmzOmajXbWPffr61Cl9CMeK1m7N2bugeMPZB40QMkLx6NxObNwZ2-vPDZ87UecMVVNLMGfLSDBv6iEnBiN_CnQrPeGFaTOXSs_)\_VerifySignature(  void \*hDeviceHandle,  int nDataType,  unsigned char \*pucData,  int nDataLength,  char \*szSignature); | | |
| 描述： | 使用设备中的可信证书验证签名 | | |
| 参数： | [in] | hDeviceHandle | 已打开的密码设备句柄 |
| [in] | nDataType | 待签名数据类型：  *0. 指令数据*  *1. 消息数据* |
| [in] | pucData | 签名数据原文 |
| [in] | nDataLength | 输入数据字节长度 |
| [in] | szSignature | 待验证签名数据 |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： | 签名值格式为 Base64( 4B Counter || 6B CertID || 64B Signature ) | | |

## 十六进制数据转换为字符串（BCD -> HEX）

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | int Hex2Ascii(  unsigned char \*pucHex,  int nHexLen,  char \*pcAscii); | | |
| 描述： | 将BCD码数据转为十六进制字符串表示 | | |
| 参数： | [in] | pucHex | 输入BCD码数据 |
| [in] | nHexLen | 输入数据字节长度 |
| [out] | pcAscii | 输出数据的16进制字符串表示 |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： | 如4字节二进制数据 0x12, 0x34, 0x56, 0x78，转换后得到“12345678” | | |

## 字符串转换为十六进制数据（HEX -> BCD）

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | int Ascii2Hex(  char \*pcAscii,  int nAaciiLen,  unsigned char \*pucHex,  int \*pnHexLen); | | |
| 描述： | 将BCD码数据转为十六进制字符串表示 | | |
| 参数： | [in] | pcAscii |  |
| [in] | nAaciiLen |  |
| [out] | pucHex |  |
| [out] | pnHexLen |  |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： | 如8位十六进制字符串“12345678”，转换后得到4字节数据 0x12, 0x34, 0x56, 0x78 | | |

## 对数据进行BASE64编码

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | unsigned char base64\_encode(  const unsigned char\* data,  int data\_len,  char\* eData,  int \*eDataLen); | | |
| 描述： | 对数据进行Base64编码 | | |
| 参数： | [in] | data | 输入待编码数据 |
| [in] | data\_len | 输入数据的长度 |
| [out] | eData | 输出编码之后的字符串数据指针 |
| [out] | eDataLen | 编码之后的字符串数据长度 |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： |  | | |

## 对数据进行BASE64解码

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | unsigned char base64\_decode(  const char\* data,  int data\_len,  unsigned char\* dData,  int \*dDataLen); | | |
| 描述： | 对Base64编码的字符串数据进行解码 | | |
| 参数： | [in] | data | 输入待解码的Base64格式字符串指针 |
| [in] | data\_len | 输入数据的长度 |
| [out] | dData | 输出解码之后的数据缓冲区指针 |
| [out] | dDataLen | 编码之后数据的字节长度 |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： |  | | |

## 计算数据摘要

|  |  |  |  |
| --- | --- | --- | --- |
| 原型： | int CalcHash(  void \*hDeviceHandle,  int nHashAlg,  unsigned char \*pucData, int nDataLength,  unsigned char \*pucHash, int \* pnHashLength); | | |
| 描述： | 对Base64编码的字符串数据进行解码 | | |
| 参数： | [in] | hDeviceHandle | 已打开的密码设备句柄 |
| [in] | nHashAlg | HASH算法标识  *1 - SM3算法（共同支持）*  *2 - SHA-1算法（共同支持）*  *4 - SHA256杂凑算法（共同支持）*  *8 - MD5杂凑算法（SJJ1507支持）*  *16 - SHA\_224杂凑算法（SJJ1507支持）*  *32 - SHA\_384杂凑算法（SJJ1507支持）*  *64 - SHA\_512杂凑算法（SJJ1507支持）* |
| [in] | pucData | 待计算数据摘要的原文数据 |
| [in] | nDataLength | 输入数据的长度 |
| [out] | pucHash | 输出HASH结果 |
| [out] | pnHashLength | HASH结果长度 |
| 返回值： | 0 | | 成功 |
| 非0 | | 失败，返回错误代码 |
| 备注： |  | | |

# JAVA版功能接口说明

## 打开密码设备

### 连接SJJ1313密码器设备（过时）

|  |  |  |
| --- | --- | --- |
| 原型： | public static void openDevice() throws Exception; | |
| 描述： | 打开SJJ1313密码器设备的连接，无参数，无返回值，若是打开设备失败，则会抛出异常。 | |
| 参数： | 无 |  |
| 返回值： | void |  |
| 异常： | Exception | 操作失败时抛出异常 |
| 备注： | 当前版本中此方法已过期，请使用带参数的功能定义 | |

### 带参数的设备连接方法

|  |  |  |
| --- | --- | --- |
| 原型： | public static void openDevice(  int flag) throws Exception; | |
| 描述： | 打开密码设备连接 | |
| 参数： | flag | 连接设备的类型标识：  *0 - 连接SJJ1507设备*  *1 - 连接SJJ1313设备* |
| 返回值： | void |  |
| 异常： | YJException | 操作失败时抛出异常  通过YJException.getErrorCode获取底层错误代码 |
| 备注： | 示例如下：  YingJGBCALLDLL.openDevice(0); // 调用sjj1507密码机  YingJGBCALLDLL.openDevice(1); // 调用sjj1313密码器 | |

## 关闭密码设备

|  |  |  |
| --- | --- | --- |
| 原型： | public static void closeDevice() throws Exception; | |
| 描述： | 关闭密码设备连接 | |
| 参数： | 无 |  |
| 返回值： | void |  |
| 异常： | YJException | 操作失败时抛出异常  通过YJException.getErrorCode获取底层错误代码 |
| 备注： | 示例如下：  YingJGBCALLDLL.closeDevice(); | |

## 计算数据签名（过时）

此版本中该方法标记为过时，建议调用方法 “[platform](http://www.baidu.com/link?url=7AqMrcWdJpnwkHbBmEIsyUn4FOKfJOQmzOmajXbWPffr61Cl9CMeK1m7N2bugeMPZB40QMkLx6NxObNwZ2-vPDZ87UecMVVNLMGfLSDBv6iEnBiN_CnQrPeGFaTOXSs_)CalculateSignature”。

|  |  |  |
| --- | --- | --- |
| 原型： | public static byte[] sign (  int dataType,  byte[] inData) throws Exception; | |
| 描述： | 计算数据签名 | |
| 参数： | dataType | 待签名数据类型：  *0. 指令数据*  *1. 消息数据* |
| inData | 待签名数据 |
| 返回值： | byte[] | 签名值数据 |
| 异常： | Exception | 操作失败时抛出异常 |
| 备注： | 当前版本中此方法已经过期 | |

## 验证数据签名（过时）

此版本中该方法标记为过时，建议调用方法 “[platform](http://www.baidu.com/link?url=7AqMrcWdJpnwkHbBmEIsyUn4FOKfJOQmzOmajXbWPffr61Cl9CMeK1m7N2bugeMPZB40QMkLx6NxObNwZ2-vPDZ87UecMVVNLMGfLSDBv6iEnBiN_CnQrPeGFaTOXSs_)VerifySignature”。

|  |  |  |
| --- | --- | --- |
| 原型： | public static boolean verifySign(  int dataType,  byte[] inData,  byte[] signature) throws Exception; | |
| 描述： | 验证数据签名 | |
| 参数： | dataType | 待签名数据类型：  *0. 指令数据*  *1. 消息数据* |
| inData | 待签名数据 |
| signature | 待验签名值数据 |
| 返回值： | boolean | true - 签名验证通过  false - 签名验证不通过 |
| 异常： | Exception | 操作过程失败时抛出异常 |
| 备注： | 当前版本中此方法已经过期 | |

## 计算数据签名（字符串形式）（过时）

此版本中该方法标记为过时，建议调用方法 “[platform](http://www.baidu.com/link?url=7AqMrcWdJpnwkHbBmEIsyUn4FOKfJOQmzOmajXbWPffr61Cl9CMeK1m7N2bugeMPZB40QMkLx6NxObNwZ2-vPDZ87UecMVVNLMGfLSDBv6iEnBiN_CnQrPeGFaTOXSs_)CalculateSignature”。

|  |  |  |
| --- | --- | --- |
| 原型： | public static String sign(  int dataType,  String inData) throws Exception; | |
| 描述： | 计算数据签名，参数以HEX字符串的形式传递 | |
| 参数： | dataType | 待签名数据类型：  *0. 指令数据*  *1. 消息数据* |
| inData | 待计算签名的数据（数据的HEX表示） |
| 返回值： | String | HEX字符串形式表示的签名值数据 |
| 异常： | Exception | 操作失败时抛出异常 |
| 备注： | 当前版本中此方法已过期 | |

## 验证数据签名（字符串形式）（过时）

此版本中该方法标记为过时，建议调用方法 “[platform](http://www.baidu.com/link?url=7AqMrcWdJpnwkHbBmEIsyUn4FOKfJOQmzOmajXbWPffr61Cl9CMeK1m7N2bugeMPZB40QMkLx6NxObNwZ2-vPDZ87UecMVVNLMGfLSDBv6iEnBiN_CnQrPeGFaTOXSs_)VerifySignature”。

|  |  |  |
| --- | --- | --- |
| 原型： | public static boolean verifySign(  int dataType,  String inData) throws Exception; | |
| 描述： | 验证数据签名，参数以HEX字符串的形式传递 | |
| 参数： | dataType | 待签名数据类型：  *0. 指令数据*  *1. 消息数据* |
| inData | 输入的签名值 |
| 返回值： | boolean | true - 签名验证通过  false - 签名验证不通过 |
| 异常： | Exception | 操作失败时抛出异常 |
| 备注： | 当前版本中此方法已过期 | |

## 计算平台数据签名

|  |  |  |
| --- | --- | --- |
| 原型： | public static String [platform](http://www.baidu.com/link?url=7AqMrcWdJpnwkHbBmEIsyUn4FOKfJOQmzOmajXbWPffr61Cl9CMeK1m7N2bugeMPZB40QMkLx6NxObNwZ2-vPDZ87UecMVVNLMGfLSDBv6iEnBiN_CnQrPeGFaTOXSs_)CalculateSignature(  int dataType,  byte[] inData) throws Exception; | |
| 描述： | 应急广播平台计算数据签名 | |
| 参数： | flag | 待签名数据类型：  *0. 指令数据*  *1. 消息数据* |
| inData | 签名数据原文 |
| 返回值： | String | Base64编码格式的签名结果  *Base64( 4B Counter || 6B CertID || 64B Signature )* |
| 异常： | YJException | 操作失败时抛出异常  通过YJException.getErrorCode获取底层错误代码 |
| 备注： | 输出签名结果格式为 Base64( 4B Counter || 6B CertID || 64B Signature )  示例如下：  //调用平台签名的方法  byte[] inData = new byte[16];  String sign = YingJGBCALLDLL.platformCalculateSignature(0, inData);  //签名结果输出格式为Base64字符串，包括4个字节的计数器，6个字节的证书编号和64个字节的签名值  AAAlUgAAAAAAQlx6kQvFG3n9iQOw3sCUrIxKlXuO1o6nVJD9iUHF0c2uWBff2Crd2JJVdG5Ay3TSAx2U5LIltdOcPjyPJcT4sFk=  //调用平台签名的方法  byte[] inData = new byte[16];  String sign = YingJGBCALLDLL.platformCalculateSignature(1, inData);  //签名结果输出格式为Base64字符串，包括4个字节的计数器，6个字节的证书编号和64个字节的签名值  AAAlfAAAAAAAQmqOMgaA92uZh47TG2etjtMVjGEPg0pXQ+ze1dvOje/00C8lE8jcnn4XX9varR/2jf9hPCIrS4mxdB3tA7shfvw= | |

## 平台验证数据签名

|  |  |  |
| --- | --- | --- |
| 原型： | public static boolean [platform](http://www.baidu.com/link?url=7AqMrcWdJpnwkHbBmEIsyUn4FOKfJOQmzOmajXbWPffr61Cl9CMeK1m7N2bugeMPZB40QMkLx6NxObNwZ2-vPDZ87UecMVVNLMGfLSDBv6iEnBiN_CnQrPeGFaTOXSs_)VerifySignature(  int dataType,  byte[] inData,  String signature) throws Exception; | |
| 描述： | 应急广播平台验证数据签名 | |
| 参数： | flag | 待签名数据类型：  *0. 指令数据*  *1. 消息数据* |
| inData | 签名数据原文 |
| signature | 待验证签名数据  *Base64( 4B Counter || 6B CertID || 64B Signature )* |
| 返回值： | boolean | true - 签名验证通过  false - 签名验证不通过 |
| 异常： | YJException | 操作失败时抛出异常  通过YJException.getErrorCode获取底层错误代码 |
| 备注： | 示例如下：  //调用平台验证签名方法  byte[] inData = new byte[16];  String sign = AAAlUgAAAAAAQlx6kQvFG3n9iQOw3sCUrIxKlXuO1o6nVJD9iUHF0c2uWBff2Crd2JJVdG5Ay3TSAx2U5LIltdOcPjyPJcT4sFk=  //返回验签结果  boolean b = YingJGBCALLDLL.platformVerifySignature(0, inData, sign);  //调用平台验证签名方法  byte[] inData = new byte[16];  String sign = AAAlfAAAAAAAQmqOMgaA92uZh47TG2etjtMVjGEPg0pXQ+ze1dvOje/00C8lE8jcnn4XX9varR/2jf9hPCIrS4mxdB3tA7shfvw=  //返回验签结果  boolean b = YingJGBCALLDLL.platformVerifySignature(1, inData, sign); | |

## 十六进制字符串数据编码（BCD -> HEX）

|  |  |  |
| --- | --- | --- |
| 原型： | public static String Hex2Ascii(  byte[] data) throws Exception; | |
| 描述： | 将字节数组形式的BCD码数据转为十六进制字符串表示 | |
| 参数： | data | 输入BCD码数据 |
| 返回值： | String | 返回数据的HEX字符串 |
| 异常： | YJException | 操作失败时抛出异常  通过YJException.getErrorCode获取底层错误代码 |
| 备注： | 如4字节二进制数据 0x12, 0x34, 0x56, 0x78，转换后得到“12345678” | |

## 十六进制字符串数据解码（HEX -> BCD）

|  |  |  |
| --- | --- | --- |
| 原型： | public static byte[] Ascii2Hex(  String inData) throws Exception; | |
| 描述： | 将十六进制字符串还原为数据的字节数组表示 | |
| 参数： | inData | 数据的HEX字符串表示 |
| 返回值： | byte[] | 数据的BCD码（字节数组） |
| 异常： | YJException | 操作失败时抛出异常  通过YJException.getErrorCode获取底层错误代码 |
| 备注： | 如8位十六进制字符串“12345678”，转换后得到4字节数据 0x12, 0x34, 0x56, 0x78 | |

## Base64编码

|  |  |  |
| --- | --- | --- |
| 原型： | public static String Base64Encode(  byte[] data) throws Exception; | |
| 描述： | 对数据进行Base64编码 | |
| 参数： | data | 输入的数据 |
| 返回值： | String | 数据的Base64编码字符串 |
| 异常： | YJException | 操作失败时抛出异常  通过YJException.getErrorCode获取底层错误代码 |
| 备注： | 示例如下：  byte[] str ={1,2,3,4,5};  String eData = YingJGBCALLDLL.Base64Encode(str);  eData显示输出为：AQIDBAU= | |

## Base64解码

|  |  |  |
| --- | --- | --- |
| 原型： | public static byte[] Base64Decode(  String edata) throws Exception; | |
| 描述： | 对Base64编码数据进行解码操作 | |
| 参数： | edata | 数据的Base64编码字符串 |
| 返回值： | byte[] | 解码后数据的字节数组 |
| 异常： | YJException | 操作失败时抛出异常  通过YJException.getErrorCode获取底层错误代码 |
| 备注： | 示例如下：  String eData = “AQIDBAU=”;  byte[] Ddata = YingJGBCALLDLL.Base64Decode(eData);  Ddata: [1, 2, 3, 4, 5] | |

## 计算数据摘要

|  |  |  |
| --- | --- | --- |
| 原型： | public static byte[] calcHash(  int hashAlg,  byte[] inData) throws Exception; | |
| 描述： | 对Base64编码的字符串数据进行解码 | |
| 参数： | hashAlg | HASH算法标识  *1 - SM3算法（共同支持）*  *2 - SHA-1算法（共同支持）*  *4 - SHA256杂凑算法（共同支持）*  *8 - MD5杂凑算法（SJJ1507支持）*  *16 - SHA\_224杂凑算法（SJJ1507支持）*  *32 - SHA\_384杂凑算法（SJJ1507支持）*  *64 - SHA\_512杂凑算法（SJJ1507支持）* |
| inData | 待计算数据摘要的原文数据 |
| 返回值： | byte[] | HASH结果 |
| 异常： | YJException | 操作失败时抛出异常  通过YJException.getErrorCode获取底层错误代码 |
| 备注： | 示例如下：  byte[] data = new byte[]{17,18,19,20};  byte[] hash = YingJGBCALLDLL.calcHash(4, data);  以十六进制输出：  Hash: 1AA2454E38E78FBE098BE157EE8112D85D6E667E2304FDE4220850FAA2F46341 | |

# 附录

SJJ1313密码器的错误码和SJJ1507的错误码都需要和应急广播指令错误码异或之后得到最终的错误码。

## SJJ1313错误代码说明

|  |  |
| --- | --- |
| **取值** | **描述** |
| 0x00000000 | 操作成功 |
| 0x01000000 | 错误码基础值 |
| 0x01000001 | 未知错误 |
| 0x01000002 | 配置文件找不到或错误 |
| 0x01000003 | 返回数据格式错误 |
| 0x01000004 | 返回数据状态字错误 |
| 0x01000005 | 打开设备失败(连不到服务) |
| 0x01000006 | 创建会话失败(连不到服务或服务拒绝) |
| 0x01000007 | 不存在的密钥调用 |
| 0x01000008 | 不支持的算法调用 |
| 0x01000009 | 不支持的算法模式调用 |
| 0x0100000A | 返回数据校验失败 |
| 0x0100000E | SM2加密数据错误 |
| 0x01000017 | 文件长度超出限制 |
| 0x01000018 | 指定的文件不存在 |
| 0x01000019 | 文件起始位置错误 |
| 0x0100001E | 密钥类型错误 |
| 0x0100001F | 密钥错误 |
| 0x01000020 | 函数执行过程中会话已经关闭 |
| 0x01000021 | 打开的会话太多 |
| 0x01000022 | 指定的会话句柄无效 |
| 0x01000025 | 私钥使用权限未获取 |
| 0x01000026 | 输入参数长度错误 |
| 0x01000027 | 指定的密钥号非法 |
| 0x01000028 | 机制无效 |
| 0x01000029 | 未调用初始化 |
| 0x0100002A | 初始化已调用 |
| 0x0100002B | 设备句柄无效 |
| 0x0100002C | 输出长度不正确 |
| 0x0100002D | 输出存储区无效 |
| 0x0100002E | 密钥存储空间已满 |
| 0x01000031 | 创建会话失败 |
| 0x01000032 | 非对称密钥的指数长度错误 |
| 0x01000033 | RSA padding 错误 |
| 0x01000034 | RSA unpadding 错误 |
| 0x01000035 | 会话密钥句柄无效 |
| 0x01000037 | 接口参数错误或不能识别 |
| 0xF0000001 | 报文不能被解析(错误报文) |
| 0xF0000002 | 打开USB设备失败(密码器掉线或故障) |
| 0xF0000003 | USB设备写失败 |
| 0xF0000004 | USB设备读失败 |
| 0xF0000040 | USB任务超时 |
| 0xF0000005 | 客户端连接数量超出上限 |
| 0xF0000006 | USB设备故障 |
| 0xF0000007 | Socket连接服务错误 |
| 0xF0000008 | socket select调用错误,在发送时 |
| 0xF0000009 | socket select调用错误,在接收时 |
| 0xF000000A | socket send错误 |
| 0xF000000B | socket recv错误 |
| 0xF000000C | socket无效 |
| 0xF000000D | socket被对方关闭 |
| 0xF000000E | socket发送超时 |
| 0xF000000F | 接收超时 |

## SJJ1507错误代码说明

|  |  |
| --- | --- |
| **错误代码** | **描述** |
| 0x00000000 | 操作成功 |
| 0x01000000 | 错误码基础值 |
| 0x01000001 | 未知错误 |
| 0x01000002 | 不支持的接口调用 |
| 0x01000003 | 与设备通信失败 |
| 0x01000004 | 运算模块无响应 |
| 0x01000005 | 打开设备失败 |
| 0x01000006 | 创建会话失败 |
| 0x01000007 | 无私钥使用权限 |
| 0x01000008 | 不存在的密钥调用 |
| 0x01000009 | 不支持的算法调用 |
| 0x0100000A | 不支持的算法模式调用 |
| 0x0100000B | 公钥运算失败 |
| 0x0100000C | 私钥运算失败 |
| 0x0100000D | 签名运算失败 |
| 0x0100000E | 验证签名失败 |
| 0x0100000F | 对称算法运算失败 |
| 0x01000010 | 多部运算步骤错误 |
| 0x01000011 | 文件长度超出限制 |
| 0x01000012 | 指定的文件不存在 |
| 0x01000013 | 文件其实位置错误 |
| 0x01000014 | 密钥类型错误 |
| 0x01000015 | 密钥错误 |
| 0x01000016 | 密文错误 |
| 0x01000017 | 随机数产生错误 |
| 0x01000018 | 私钥使用权限获取失败 |
| 0x01000019 | MAC运算失败 |
| 0x0100001A | 指定的文件已经存在 |
| 0x0100001B | 文件写错误 |
| 0x0100001C | 存储空间不足 |
| 0x0100001D | 输入参数错误 |
| 0x0100001E | 输出参数错误 |
| 0x0100001F | 函数执行过程中，该会话已经关闭 |
| 0x01000020 | 打开会话过多 |
| 0x01000021 | 指定会话句柄无效 |
| 0x01000022 | 获取私钥使用权限失败 |
| 0x01000023 | 获取私钥使用权限重复 |
| 0x01000024 | 私钥使用权限未获取 |
| 0x01000025 | 输入参数长度只是错误 |
| 0x01000026 | 指定密钥号非法 |
| 0x01000027 | 机制无效 |
| 0x01000028 | 未调用初始化 |
| 0x01000029 | 初始化已调用 |
| 0x0100002A | 设备句柄无效 |
| 0x0100002B | 设备异常 |
| 0x0100002C | 密钥存储空间已满 |
| 0x0100002D | 产生密钥失败 |
| 0x0100002F | 密钥不存在 |
| 0x01000030 | 内存分配失败 |
| 0x01000031 | 数据不允许 |
| 0x01000032 | ECC密钥长度错误 |
| 0x01000033 | 会话初始化失败 |
| 0x01000034 | 产生随机数失败 |
| 0x01000035 | 非对称密钥指数长度错误 |
| 0x01000036 | RSA padding错误 |
| 0x01000037 | RSA unpadding错误 |
| 0x01000038 | 会话密钥句柄无效 |
| 0x01000039 | 读取配置文件失败 |
| 0x01000040 | 连接密码机心跳测试失败 |
| 0x01000041 | 数据传输错误 |
| 0x01000042 | 获取设备信息失败 |
| 0x01000043 | 随机数生成失败 |
| 0x01000044 | 索引超出范围 |
| 0x01000045 | 输入句柄无效 |
| 0x01000046 | 获取内部私钥权限失败 |
| 0x01000047 | 导出公钥失败 |
| 0x01000048 | 生成会话密钥失败 |
| 0x01000049 | 数字信封转换失败 |
| 0x01000050 | 生成协商密钥失败 |
| 0x01000051 | 导入RSA密钥对失败 |
| 0x01000052 | 导入ECC密钥对失败 |
| 0x01000053 | ECC公钥加密失败 |
| 0x01000054 | IV数据错误 |
| 0x01000055 | 杂凑运算初始化失败 |
| 0x01000056 | 杂凑运算失败 |
| 0x01000057 | 创建文件失败 |
| 0x01000058 | 删除文件失败 |

## 应急广播命令错误码说明

|  |  |  |
| --- | --- | --- |
| **宏定义** | **错误代码** | **描述** |
| YJERR\_OK | 0x00000000 | 操作成功 |
| YJERR\_OPENDEVICE | 0x00000100 | 打开设备失败 |
| YJERR\_CLOSEDEVICE | 0x00000200 | 关闭设备失败 |
| YJERR\_OPENSESSION | 0x00000300 | 打开会话失败 |
| YJERR\_CLOSESESSION | 0x00000400 | 关闭会话失败 |
| YJERR\_CREATEFILE | 0x00000500 | 创建文件失败 |
| YJERR\_DELETEFILE | 0x00000600 | 删除文件失败 |
| YJERR\_READFILE | 0x00000700 | 读取文件失败 |
| YJERR\_WRITEFILE | 0x00000800 | 写入文件失败 |
| YJERR\_HASHINIT | 0x00000900 | HASH运算初始化操作失败 |
| YJERR\_HASHUPDATE | 0x00000A00 | HASH运算过程操作失败 |
| YJERR\_HASHFINAL | 0x00000B00 | HASH运算结束操作失败 |
| YJERR\_EXPPUBKEY | 0x00000C00 | 导出公钥失败 |
| YJERR\_ACCESSPRIKEY | 0x00000D00 | 获取私钥权限失败 |
| YJERR\_RELEASESPRIKEY | 0x00000E00 | 释放私钥权限失败 |
| YJERR\_SIGNATURE | 0x00000F00 | 签名运算失败 |
| YJERR\_VERIFY | 0x00001000 | 验签运算失败 |
| YJERR\_FILEOPEN | 0x0000E100 | 打开本机文件失败 |
| YJERR\_FILEREAD | 0x0000E200 | 读取本机文件失败 |
| YJERR\_COUNTERMUTEXFAILED | 0x0000E300 | 签名计数器同步异常 |
| YJERR\_COUNTER | 0x0000E400 | 签名计数器操作失败 |
| YJERR\_INDEXMUTEXFAILED | 0x0000E500 | 证书索引文件同步异常 |
| YJERR\_INDEX | 0x0000E600 | 证书索引文件操作失败 |
| YJERR\_MEMSMALL | 0x0000E700 | 提供的缓冲区空间过小 |
| YJERR\_CERTINDEX | 0x0000E800 | 证书索引超限 |
| YJERR\_NOCERT | 0x0000E900 | 证书不存在（或无证书） |
| YJERR\_CERTCOUNTMAX | 0x0000EA00 | 证书数量超限 |
| YJERR\_STRLEN | 0x0000EB00 | 字符串长度错误（可能要求为偶数） |
| YJERR\_STRVALUE | 0x0000EC00 | 字符串数据错误（可能要求十六进制字符串或Base64编码） |