****

|  |  |
| --- | --- |
| **保密级别** | 项目内部公开 |
| **版 本 号** | V1.0 |
| **文档编号** |  |

**宝能汽车PKI**

**安全平台SDK接口文档**

**撰写部门:**IOT业务开发部

**撰写时间:**2020年9月14日

**发行范围:**宝能汽车PKI项目组

**文档记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **版本编号** | **变化状态** | **简要说明** | **撰写/变更人** | **批准人** | **批准日期** |
| V1.0 | C | 初稿 | 刘涛 |  | 2020-9-14 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

\*变化状态:C――创建、A――增加、M――修改（+修改说明）、D――删除（+删除说明）

目录

[目录 2](#_Toc51074862)

[1. 文档说明 3](#_Toc51074863)

[2. SDK初始化接口 3](#_Toc51074864)

[3. 证书管理类接口 4](#_Toc51074865)

[3.1. 在线证书申请接口 4](#_Toc51074866)

[3.2. 证书吊销接口 5](#_Toc51074867)

[3.3. 证书查询接口 6](#_Toc51074868)

[3.4. 离线证书申请 7](#_Toc51074869)

[4. 远程控制类接口 8](#_Toc51074870)

[4.1. 对称加密接口 8](#_Toc51074871)

[4.2. 对称解密接口 9](#_Toc51074872)

[4.3. 数据签名接口 9](#_Toc51074873)

[4.4. 数据验签接口 10](#_Toc51074874)

[5. OTA业务相关接口 11](#_Toc51074875)

[5.1. OTA文件加密接口 11](#_Toc51074876)

[6. 数字钥匙业务相关接口 12](#_Toc51074877)

[6.1. 数字钥匙数据加密接口 12](#_Toc51074878)

[7. CRL同步接口 13](#_Toc51074879)

[7.1. 获取crl接口 13](#_Toc51074880)

[8. 附录 14](#_Toc51074881)

[8.1. AbstractResult类 14](#_Toc51074882)

[8.2. SdkConfig类 15](#_Toc51074883)

[8.3. OfflineCertBean类 16](#_Toc51074884)

[8.4. IssueCertRequest类 18](#_Toc51074885)

# 文档说明

本文档是为TSP平台对接PKI平台提供的安全SDK接口说明，依据需求和技术方案，该文档主要包含SDK初始化接口、证书管理类接口、远程控制类接口、OTA业务相关接口、数字钥匙业务相关接口这几个方面。SDK基于java语言开发，以jar包+动态库的方式提供给TSP业务平台调用，SDK各接口返回值均按照java对象的方式返回，便于扩展，各对象都包含code和msg两个属性，其中code表示成功或失败的状态码，取值为0表示调用成功，其他取值参见各对象属性定义，msg表示成功或失败的原因描述。

# SDK初始化接口

|  |  |
| --- | --- |
| 接口名称 | SDK初始化接口 |
| 功能描述 | 实现SDK的初始化的工作 |
| 调用方式 | java |
| 接口定义 | InitResult init(SdkConfig sdkConfig); |
| 接口参数说明 | /\*\*  \* sdk初始化接口  \* @param sdkConfig sdk配置类，参考附录[SdkConfig](#_SdkConfig类)类  \* @return InitResult  \*/ |
| 返回值说明 | package com.xdja.iot.pki.sdk.bean;  /\*\*  \* SDK初始化结果  \*/  public final class InitResult extends [AbstractResult](#_AbstractResult类) {  /\*\*  \* 解析rsakeystore失败  \*/  public static final Integer PARSE\_RSAKEYSTORE\_FAIL = 12;  /\*\*  \* 服务端验证sdk证书失败  \*/  public static final Integer VALID\_RSAKEYSTORE\_FAIL = 13;  /\*\*  \* 软件密码模块初始化失败  \*/  public static final Integer VHSM\_INIT\_FAIL = 14;  } |

# 证书管理类接口

## 在线证书申请接口

|  |  |
| --- | --- |
| 接口名称 | 在线证书申请 |
| 功能描述 | 实现证书在线申请的功能 |
| 调用方式 | java |
| 接口定义 | IssueCertResult issueCert(String terminalId,String terminalType,String csr,Integer certValidity,Integer algorithm); |
| 接口参数说明 | /\*\*  \* 在线申请证书  \* @param terminalId 要颁发证书的设备唯一标识（长度不超过64个字符）  \* @param terminalType 终端类型 1-tbox 2-蓝牙  \* @param csr 证书签发请求（要求使用PKCS#10格式）  \* @param certValidity 证书有效期(单位：天)  \* @param algorithm 算法标识，取值：1表示rsa算法  \* @return IssueCertResult  \*/ |
| 返回值说明 | package com.xdja.iot.pki.sdk.bean;  /\*\*  \* 在线申请证书结果  \*/  public final class IssueCertResult extends [AbstractResult](#_AbstractResult类) {  /\*\*  \* 待颁发的证书已经存在  \*/  public static final Integer CERT\_EXISTED = 21;  /\*\*  \* PKI服务端异常  \*/  public static final Integer PKI\_SERVER\_ERROR = 22;  String cert;  public String getCert() {  return cert;  }  public void setCert(String cert) {  this.cert = cert;  }  } |

## 证书吊销接口

|  |  |
| --- | --- |
| 接口名称 | 证书吊销 |
| 功能描述 | 实现证书吊销的功能 |
| 调用方式 | java |
| 接口定义 | RevokeCertResult revokeCert(String terminalId,Integer algorithm); |
| 接口参数说明 | /\*\*  \* 撤销证书  \* @param terminalId 要撤销证书的设备唯一标识（长度不超过64个字符）  \* @param algorithm algorithm 算法标识，取值：1表示rsa算法  \* @return RevokeCertResult  \*/ |
| 返回值说明 | package com.xdja.iot.pki.sdk.bean;  /\*\*  \* 撤销证书结果  \*/  public final class RevokeCertResult extends [AbstractResult](#_AbstractResult类) {  /\*\*  \* 待撤销设备的证书不存在  \*/  public static final Integer CERT\_NOT\_EXISTED = 31;  /\*\*  \* 撤销证书失败  \*/  public static final Integer REVOKE\_CERT\_FAIL = 35;  } |

## 证书查询接口

|  |  |
| --- | --- |
| 接口名称 | 证书查询 |
| 功能描述 | 实现证书查询的功能 |
| 调用方式 | java |
| 接口定义 | QueryCertResult queryCert(String terminalId,Integer algorithm); |
| 接口参数说明 | /\*\*  \* 查询证书  \* @param terminalId 要查询证书的设备唯一标识（长度不多余64个字符）  \* @param algorithm algorithm 算法标识，取值：1表示rsa算法  \* @return RevokeCertResult  \*/ |
| 返回值说明 | package com.xdja.iot.pki.sdk.bean;  /\*\*  \* 查询证书结果  \*/  public final class QueryCertResult extends [AbstractResult](#_AbstractResult类) {  /\*\*  \* 待查询的设备的证书不存在  \*/  public static final Integer CERT\_NOT\_EXISTED = 41;  /\*\*  \* 查询证书失败  \*/  public static final Integer QUERY\_CERT\_FAIL = 45;  /\*\*  \* pem格式，base64字符串  \*/  String cert;  public String getCert() {  return cert;  }  public void setCert(String cert) {  this.cert = cert;  }  } |

## 离线证书申请

|  |  |
| --- | --- |
| 接口名称 | 证书更新 |
| 功能描述 | 实现证书更新的功能 |
| 调用方式 | java |
| 接口定义 | IssueCertOfflineResult issueCertOffline(List<[IssueCertRequest](#_IssueCertRequest类)> issueCertOfflineRequests); |
| 接口参数说明 | /\*\*  \* 离线证书申请（用于产线证书罐装）  \* @param issueCertOfflineRequests 离线证书申请请求  \* @return IssueCertOfflineResult  \*/ |
| 返回值说明 | package com.xdja.iot.pki.sdk.bean;  import java.util.List;  /\*\*  \* 离线申请证书结果  \*/  public final class IssueCertOfflineResult extends [AbstractResult](#_AbstractResult类) {  /\*\*  \* 离线申请证书失败  \*/  public static final Integer ENCRYPT\_FAIL = 305;  /\*\*  \* 离线证书列表  \*/  List<[OfflineCertBean](#_OfflineCertBean类)> offlineCertList ;  public List<OfflineCertBean> getOfflineCertList() {  return offlineCertList;  }  public void setOfflineCertList(List<OfflineCertBean> offlineCertList) {  this.offlineCertList = offlineCertList;  }  } |

# 远程控制类接口

## 对称加密接口

|  |  |
| --- | --- |
| 接口名称 | 对称加密接口 |
| 功能描述 | 实现数据的加密功能，支持AES算法 |
| 调用方式 | java |
| 接口定义 | EncryptResult encrypt(byte[] content,byte[] key,Integer algorithm); |
| 接口参数说明 | /\*\*  \* 对称加密  \* @param content 待加密内容字节数组  \* @param key 加密密钥字节数组  \* @param algorithm 加密算法 取值 1表示AES  \* @return EncryptResult  \*/ |
| 返回值说明 | package com.xdja.iot.pki.sdk.bean;  /\*\*  \* 对称加密结果  \*/  public final class EncryptResult extends [AbstractResult](#_AbstractResult类) {  /\*\*  \* 加密失败  \*/  public static final Integer ENCRYPT\_FAIL = 55;  /\*\*  \* 加密后的内容  \*/  byte[] content;  public byte[] getContent() {  return content;  }  public void setContent(byte[] content) {  this.content = content;  }  } |

## 对称解密接口

|  |  |
| --- | --- |
| 接口名称 | 对称解密接口 |
| 功能描述 | 实现数据的对称解密功能，支持AES算法 |
| 调用方式 | java |
| 接口定义 | DecryptResult decrypt(byte[] cipherText,byte[] key,Integer algorithm); |
| 接口参数说明 | /\*\*  \* 对称解密  \* @param cipherText 待解密的密文  \* @param key 解密密钥字节数组  \* @param algorithm 解密算法 取值 1表示AES  \* @return DecryptResult  \*/ |
| 返回值说明 | package com.xdja.iot.pki.sdk.bean;  /\*\*  \* 对称解密结果  \*/  public final class DecryptResult extends [AbstractResult](#_AbstractResult类) {  /\*\*  \* 解密失败  \*/  public static final Integer DECRYPT\_FAIL = 65;  byte[] content;  public byte[] getContent() {  return content;  }  public void setContent(byte[] content) {  this.content = content;  }  } |

## 数据签名接口

|  |  |
| --- | --- |
| 接口名称 | 数据签名 |
| 功能描述 | 实现数据的签名功能 |
| 调用方式 | java |
| 接口定义 | SignResult signMessage(String applicationName,byte[] inData,Integer type); |
| 接口参数说明 | /\*\*  \* 数据签名  \* @param applicationName 应用系统名称 取值：ota或bluetooth  \* @param inData 签名的数据  \* @param type 取值1表示传入的inData是数据原文；2表示传入的inData是数据原文的hash  \* @return SignResult  \*/ |
| 返回值说明 | package com.xdja.iot.pki.sdk.bean;  /\*\*  \* 数据签名结果  \*/  public final class SignResult extends [AbstractResult](#_AbstractResult类) {  /\*\*  \* 签名失败  \*/  public static final Integer SIGN\_FAIL = 75;  /\*\*  \* base64字符串  \*/  String signValue;  } |

## 数据验签接口

|  |  |
| --- | --- |
| 接口名称 | 数据验签 |
| 功能描述 | 实现数据的验签功能 |
| 调用方式 | java |
| 接口定义 | VerifySignDataResult verifySignMessage(String applicationName,byte[] inData,String signValue,Integer type); |
| 接口参数说明 | /\*\*  \* 数据验签  \* @param applicationName 应用系统名称 取值：ota或bluetooth  \* @param inData 待签证的数据  \* @param signValue 签名值  \* @param type 取值1表示传入的inData是数据原文；2表示传入的inData是数据原文的hash  \* @return VerifySignDataResult  \*/ |
| 返回值说明 | package com.xdja.iot.pki.sdk.bean;  /\*\*  \* 数据验签  \*/  public final class VerifySignDataResult extends [AbstractResult](#_AbstractResult类) {  /\*\*  \* 验证签名失败  \*/  public static final Integer VERIFY\_ERROR = 85;  } |

# OTA业务相关接口

## OTA文件加密接口

|  |  |
| --- | --- |
| 接口名称 | OTA文件加密 |
| 功能描述 | 实现OTA文件的加密 |
| 调用方式 | java |
| 接口定义 | OtaFileEnCryptResult encryptOtaFile(String otaFileInputPath,String otaFileVersion,String otaFileOutputPath); |
| 接口参数说明 | /\*\*  \* ota文件加密  \* @param otaFileInputPath 待加密的ota文件路径  \* @param otaFileVersion ota文件版本号  \* @param otaFileOutputPath 加密后的ota文件路径  \* @return OtaFileEnCryptResult  \*/ |
| 返回值说明 | package com.xdja.iot.pki.sdk.bean;  /\*\*  \* ota文件加密结果  \*/  public final class OtaFileEnCryptResult extends [AbstractResult](#_AbstractResult类) {  /\*\*  \* OTA文件加密失败  \*/  public static final Integer ENCRYPT\_ERROR = 95;  } |

# 数字钥匙业务相关接口

## 数字钥匙数据加密接口

|  |  |
| --- | --- |
| 接口名称 | 数字钥匙数据加密 |
| 功能描述 | 实现数字钥匙数据的加密 |
| 调用方式 | java |
| 接口定义 | BluetoothFileEnCryptResult encryptBluetoothContent(String terminalId,byte[] content,Integer algorithm); |
| 接口参数说明 | /\*\*  \* 蓝牙数据加密,采用非对称加密算法  \* @param terminalId 设备唯一标识（长度不多余64个字符），必填字段  \* @param content 待加密数据内容字节数组  \* @param algorithm 加密算法 取值 1表示RSA  \* @return BluetoothFileEnCryptResult  \*/ |
| 返回值说明 | package com.xdja.iot.pki.sdk.bean;  /\*\*  \* 蓝牙数据加密结果  \*/  public final class BluetoothFileEnCryptResult extends [AbstractResult](#_AbstractResult类) {  /\*\*  \* 蓝牙文件加密失败  \*/  public static final Integer ENCRYPT\_FAIL = 105;  /\*\*  \* 加密后的密文  \*/  private byte[] cipherText;  public byte[] getCipherText() {  return cipherText;  }  public void setCipherText(byte[] cipherText) {  this.cipherText = cipherText;  }  } |

# CRL同步接口

## 获取crl接口

|  |  |
| --- | --- |
| 接口名称 | 获取crl |
| 功能描述 | 实现crl获取 |
| 调用方式 | java |
| 接口定义 | CrlResult getCrls(); |
| 接口参数说明 | /\*\*  \* 获取crl接口  \* @return CrlResult  \*/ |
| 返回值说明 | package com.xdja.iot.pki.sdk.bean;  import java.security.cert.X509CRL;  import java.util.List;  /\*\*  \* crl同步结果类  \*/  public final class CrlResult extends [AbstractResult](#_AbstractResult类){  /\*\*  \* 拉去CRL失败  \*/  public static final Integer ENCRYPT\_FAIL = 205;  /\*\*  \* crl列表  \*/  List<X509CRL> crlList;  public List<X509CRL> getCrlList() {  return crlList;  }  public void setCrlList(List<X509CRL> crlList) {  this.crlList = crlList;  }  } |

# 附录

## AbstractResult类

package com.xdja.iot.pki.sdk.bean;

abstract class AbstractResult {

/\*\*

\* 调用成功状态码，适用于所有接口

\*/

public static final Integer SUCCESS = 0;

/\*\*

\* 传入参数不合法

\*/

public static final Integer PARAM\_ERROR=1001;

/\*\*

\* 连接PKI server异常

\*/

public static final Integer CONNECT\_PKI\_ERROR=1002;

private int code;

private String msg;

public int getCode() {

return code;

}

public void setCode(int code) {

this.code = code;

}

public String getMsg() {

return msg;

}

public void setMsg(String msg) {

this.msg = msg;

}

}

## SdkConfig类

package com.xdja.iot.pki.sdk.bean;

public class SdkConfig {

/\*\*

\* pki server ip

\*/

String pkiServerIp;

/\*\*

\* pki server port

\*/

Integer pkiServerPort;

/\*\*

\* sdk rsa keystore path

\*/

String RsaKeyStorePath;

/\*\*

\* sdk rsa keystore password

\*/

String RsaKeyStorePassword;

/\*\*

\* 软件密码模块工作路径

\*/

String vhsmWorkPath;

public String getPkiServerIp() {

return pkiServerIp;

}

public void setPkiServerIp(String pkiServerIp) {

this.pkiServerIp = pkiServerIp;

}

public Integer getPkiServerPort() {

return pkiServerPort;

}

public void setPkiServerPort(Integer pkiServerPort) {

this.pkiServerPort = pkiServerPort;

}

public String getRsaKeyStorePath() {

return RsaKeyStorePath;

}

public void setRsaKeyStorePath(String rsaKeyStorePath) {

RsaKeyStorePath = rsaKeyStorePath;

}

public String getRsaKeyStorePassword() {

return RsaKeyStorePassword;

}

public void setRsaKeyStorePassword(String rsaKeyStorePassword) {

RsaKeyStorePassword = rsaKeyStorePassword;

}

public String getVhsmWorkPath() {

return vhsmWorkPath;

}

public void setVhsmWorkPath(String vhsmWorkPath) {

this.vhsmWorkPath = vhsmWorkPath;

}

}

## OfflineCertBean类

package com.xdja.iot.pki.sdk.bean;

import java.io.Serializable;

/\*\*

\* 离线证书实体类

\*/

public class OfflineCertBean implements Serializable {

/\*\*

\* 证书内容，pem格式，base64字符串

\*/

String cert;

/\*\*

\* 证书私钥，pem格式，base64字符串

\*/

String privateKey;

/\*\*

\* 设备唯一标识

\*/

String terminalId;

/\*\*

\* 设备类型

\*/

Integer terminalType;

/\*\*

\* 证书算法

\*/

Integer algorithm;

public String getCert() {

return cert;

}

public void setCert(String cert) {

this.cert = cert;

}

public String getPrivateKey() {

return privateKey;

}

public void setPrivateKey(String privateKey) {

this.privateKey = privateKey;

}

public String getTerminalId() {

return terminalId;

}

public void setTerminalId(String terminalId) {

this.terminalId = terminalId;

}

public Integer getTerminalType() {

return terminalType;

}

public void setTerminalType(Integer terminalType) {

this.terminalType = terminalType;

}

public Integer getAlgorithm() {

return algorithm;

}

public void setAlgorithm(Integer algorithm) {

this.algorithm = algorithm;

}

}

## IssueCertRequest类

package com.xdja.iot.pki.sdk.bean;

import java.io.Serializable;

public class IssueCertRequest implements Serializable {

String terminalId;

/\*\*

\* 设备类型 1-tbox 2-蓝牙

\*/

Integer terminalType;

/\*\*

\* 算法表示 1-rsa

\*/

Integer algorithm;

/\*\*

\* 证书有效期（单位：天）

\*/

Integer certValidity;

public String getTerminalId() {

return terminalId;

}

public void setTerminalId(String terminalId) {

this.terminalId = terminalId;

}

public Integer getTerminalType() {

return terminalType;

}

public void setTerminalType(Integer terminalType) {

this.terminalType = terminalType;

}

public Integer getAlgorithm() {

return algorithm;

}

public void setAlgorithm(Integer algorithm) {

this.algorithm = algorithm;

}

public Integer getCertValidity() {

return certValidity;

}

public void setCertValidity(Integer certValidity) {

this.certValidity = certValidity;

}

}