CEN

WORKSHOP

AGREEMENT

CWA 14923-2

May 2004

ICS 35.240.40

Supersedes CWA 13937-2:2003

English version

J/eXtensions for Finalcial Sevices (J/XFS) for the Java Platform - Part 2: Pin Keypad Device Class Interface - Programmer's Reference

EUROPEAN COMMITTEE FOR STANDARDIZATION
COMITÉ EUROPÉEN DE NORMALISATION
EUROPÄISCHES KOMITEE FÜR NORMUNG

Management Centre: rue de Stassart, 36   B-1050 Brussels

Contents

FOREWORD

This CWA contains the specifications that define the J/eXtensions for Financial Services (J/XFS) for the Java ™ Platform, as developed by the J/XFS Forum and endorsed by the CEN/ISSS J/XFS Workshop. J/XFS provides an API for Java applications which need to access financial devices. It is hardware independent and, by using 100% pure Java, also operating system independent.

The CEN/ISSS J/XFS Workshop gathers suppliers (among others the J/XFS Forum members), service providers as well as banks and other financial service companies. A list of companies participating in this Workshop and in support of this CWA is available from the CEN/ISSS Secretariat. The specification was agreed upon by the J/XFS Workshop Meeting of 2004-02-10/11 in Saint-Denis (Paris) and a subsequent electronic review by the Workshop participants, and the final version was sent to CEN for publication on 2004-03-24.

The specification is continuously reviewed and commented in the CEN/ISSS J/XFS Workshop. The information published in this CWA is furnished for informational purposes only. CEN/ISSS makes no warranty expressed or implied, with respect to this document. Updates of the specification will be available from the CEN/ISSS J/XFS Workshop public web pages pending their integration in a new version of the CWA (see: http://www.cenorm.be/cenorm/businessdomains/businessdomains/informationsocietystandardizationsystem/applying+technologies/j-xfs+workshop/index.asp).

The J/XFS specifications are now further developed in the CEN/ISSS J/XFS Workshop. CEN/ISSS Workshops are open to all interested parties offering to contribute. Parties interested in participating should contact the CEN/ISSS Secretariat (isss@cenorm.be). To submit questions and comments for the J/XFS specifications, please contact the J/XFS Workshop Secretariat hosted in CEN/ISSS (jxfs-helpdesk@cenorm.be).

Questions and comments can also be submitted to the members of the J/XFS Forum, who are all CEN/ISSS J/XFS Workshop members, through the J/XFS Forum web-site http://www.jxfs.com

This CWA is composed of the following parts:

- Part 1: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Base Architecture - Programmer's Reference
- Part 2: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Pin Keypad Device Class Interface - Programmer's Reference
- Part 3: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Magnetic Stripe & Chip Card Device Class Interface - Programmer's Reference
- Part 4: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Text Input/Output Device Class Interface - Programmer's Reference
- Part 5: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Cash Dispenser, Recycler and ATM Interface - Programmer's Reference
- Part 6: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Printer Device Class Interface - Programmer's Reference
- Part 7: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Alarm Device - Programmer's Reference
- Part 8: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Sensors and Indicators Unit Device Class Interface - Programmer's Reference
- Part 9: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Depository Device Class Interface - Programmer's Reference
- Part 10: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Check Reader/Scanner Device Class Interface - Programmer's Reference
- Part 11: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Camera Specification - Programmer's Reference
- Part 12: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Vendor Dependant Mode Specification - Programmer's Reference

CWA 14923-2:2004 replaces CWA 13937-2:2003 and should be read in conjunction with CWA 13937-2:2000, which contains the previous release of the J/XFS specification

Note: Java and all Java-based trademarks and logos are trademarks of Sun Microsystems, Inc. The Java Trademark Guidelines are currently available on the web at http://java.sun.com/nav/business/trademark_guidelines.html.
All other trademarks are trademarks of their respective owners.

HISTORY

The main differences to the previous CWA 13937:2000 are:

- Included the functions and features regarding the Remote Key loading.
- Added properties kuseRSAPrivateand kuseRSAPrivateSign in the JxfsPINKeyUses class
- Included the GENAS protocol
- A new method is created specially dedicated to import a RSA public key : importEMVRSAPublicKey.
- A new class JxfsPINEMVRSAKeyToImport is created.
- JxfsRSASignatureAlgo is replaced by JxfsPINRSAIntegrityAlgorithm.
- JxfsPINEMVCryptoModes data class is created
- JxfsPINKeyUses data class, added a new property : KuseRSAPublicKeyVerifiy and clarified the description of kusRSAPublicKey.
- JxfsPINBlockData data class : added clarification when it is used for EMV
- added a new method to delete a key from the encryption module: deleteKey
- JxfsPINKeyVerificationData added clarification in the keyVerCode property
- Added clarification in the Initialize method
- Added an Appendix to defines the EMV requirements and clarifications
- Added EMV features:
- Removed getBMP / setBMP methods from JxfsPINSecureMsgISO
- Added the ZKA extension
- Added changes following the proposal on the read methods of document 2001/037
- Added JxfsPINReadMode2 support class
- Added readData(JxfsPINReadMode2 readMode) to IJxfsPINKeypadControl
- Added secureReadPin(JxfsPINReadMode2 readMode) to IJxfsPINKeypadControl
- Added the eventOnStartSupported property in the IJxfsPINKeypadControl
- JXFS_E_CLAIMED exception removed from section 4.2
- keyEncKey property of JxfsPINCryptoData (section 5.29.1) changed from String to byte[]
- Added a class hierarchy diagram
- Added 2 header pages: title and history
- Added paragraph describing handling of null parameters

1 Scope

This document describes the Pin Keypad Device (PIN) classes based on the basic architecture of J/XFS which is similar to the JavaPOS architecture. It is event driven and asynchronous.

Three basic levels are defined in JavaPOS. For J/XFS this model is extended by a communication layer, which provides device communication that allows distribution of applications and devices within a network. So we have the following layers in J/XFS :

- Application
- Device Control and Device Manager
- Device Communication
- Device Service

Application developers program against control objects and the Device Manager which reside in the Device Control layer. This is the usual interface between applications and J/XFS devices. Device Control objects access the Device Manager to find an associated Device Service. Device Service objects provide the functionality to access the real device (i.e. like a device driver).
During application startup the Device Manager is responsible for locating the desired Device Service object and attaching this to the requesting Device Control object. Location and/or routing information for the Device Manager reside in a central repository.

To support Pin Keypad devices the basic Device Control structure is extended with various properties and methods specific to this device which are described on the following pages.

2 Overview

2.1 Description

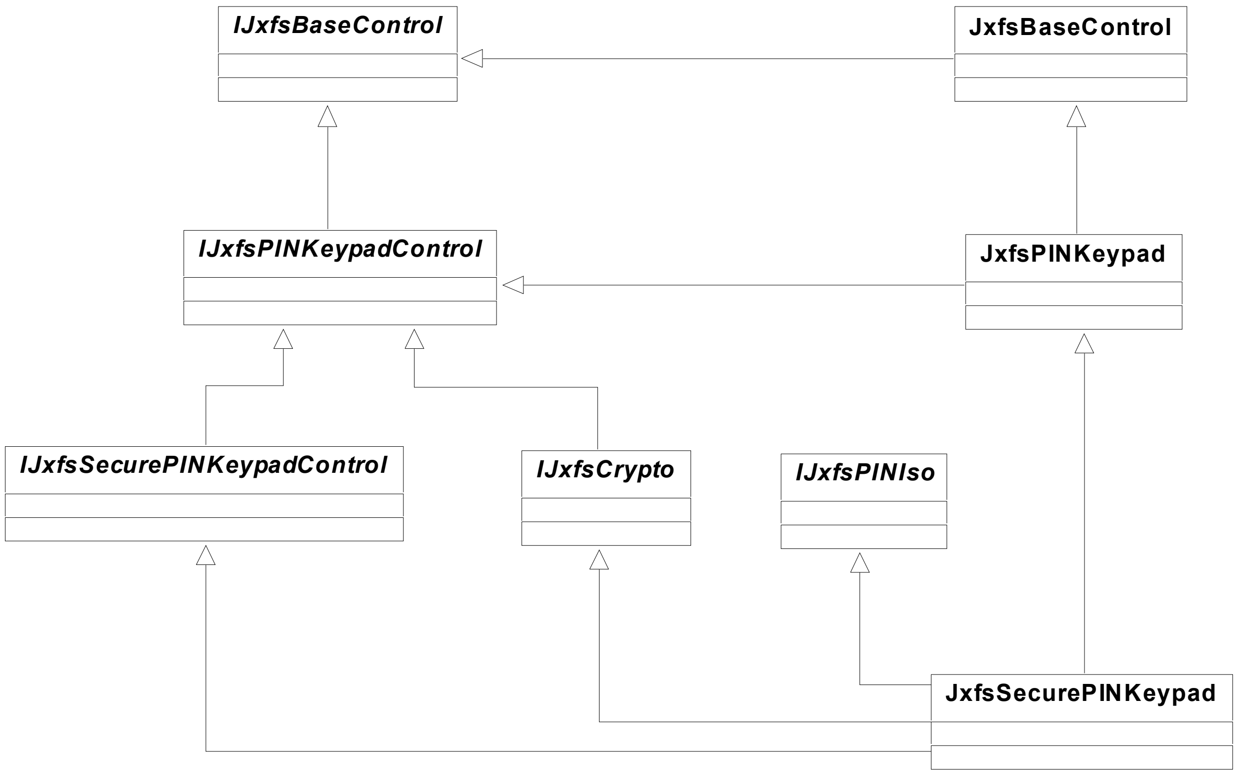
This specification covers the interfaces and classes to access personal identification keypads (PIN pads). The main functions of PIN Keypad devices supported in this specification are:

- Non secure key pad functions (like key press detection, plain PIN retrieval,...)
- Secure PIN operations (like PIN validation, data encryption with PIN as cryptography input,...)
- Cryptographic services (like data encryption/decryption, MAC generation,...)

The J/XFS PIN Keypad specification separates the PIN Keypad functions between generic non-secure keypad functions and security-related functions, that is, the ones related to cryptography.

As well as the rest of J/XFS device controls, the J/XFS PIN Keypad Device Support uses the event driven model and the same behavioral model. Therefore, the application will instantiate a J/XFS PIN Keypad Device Control Object and then use the available methods to do I/O. When an I/O method is called, the J/XFS PIN Keypad Device Service will attempt to process the requested I/O. If the request is invalid or an exception is encountered, the application will be notified by a J/XFS exception. Completion of the request will be reported by an event. Thus the application must register itself with the J/XFS PIN Keypad Device Control Object for the various types of events it wishes to handle.

2.2 Class Hierarchy



The IJxfsPINIso interface is mandatory for the JxfsSecurePINKeypad control, but is optional for the implementing device service (see Appendix A).

2.3 Classes and Interfaces

The following classes and interfaces are used by the J/XFS PIN Keypad Device Controls.

Class or Interface	Name	Description	Extends / Implements
Interface	**IJxfsBaseControl**	Base interface for all the device controls. Contains methods common to all the device controls.	--
Interface	**IJxfsPINKeypadControl**	Base interface for PIN controls. Contains methods declarations specific to PIN device controls.	Extends: **IJxfsBaseControl**
Interface	**IJxfsSecurePINKeypadControl**	Interface for PIN controls implementing secure PIN entry and validation. Contains methods specific to device controls for the secure PIN device category.	Extends: **IJxfsPINKeypadControl**
Interface	**IJxfsCrypto**	Interface for PIN controls implementing security and cryptographic functions.	Extends: **IJxfsPINKeypadControl**
Class	**JxfsBaseControl**	Base class for all the device controls. Contains properties common to all the device controls.	
Class	**JxfsPINKeypad**	Base class for PIN controls. Contains properties specific to PIN device controls.	Implements: **IJxfsPINKeypadControl**
Class	**JxfsSecurePINKeypad**	Class for PIN controls implementing security and cryptographic functions.	Extends: **JxfsPINKeypad** Implements: **IJxfsSecurePINKeypadControl, IJxfsCrypto**

2.4 Support Classes

Class or Interface	**Name**	**Description**	**Extends / Implements**
Interface	**JxfsConst**	Interface containing the Jxfs constants that are common to several device categories	--
Interface	**JxfsPINConst**	Interface containing the Jxfs constants that are common to all the PIN device controls.	**--**
Class	**JxfsPINFKeySet**	PIN function keys selector class. Indicates for each function key if it is selected or not. Properties are read only.	Extends: **JxfsType**
Class	**JxfsPINFKeysSelection**	Subclass of JxfsPINFKeySet. It contains the same properties, but they can be set by applications.	Extends: **JxfsPINFKeySet**
Class	**JxfsPINFDKeysSelection**	PIN function descriptor keys selector class. Indicates for each function descriptor key if it is selected or not.	Extends: **JxfsType**
Class	**JxfsPINFDKey**	Data class that contains information about a function descriptor key (FDKey).	Extends: **JxfsType**
Class	**JxfsPINReadMode**	Data class that defines the conditions for PIN keypad input operations.	Extends: **JxfsType**
Class	**JxfsPINReadMode2**	Data class that defines extended conditions for PIN keypad input operations	Extends: **JxfsPINReadMode**
Class	**JxfsPINPressedKey**	Data class that contains information about a key pressed during an input operation.	Extends: **JxfsType**
Class	**JxfsPINReadData**	Data class that contains the information provided to the application when an input operation completes.	Extends: **JxfsType**
Class	**JxfsPINFormats**	PIN formats selector class. Indicates for each PIN format if it is selected or not. Properties are read only.	Extends: **JxfsType**
Class	**JxfsPINValidationAlgorit hms**	PIN validation algorithms selector class. Indicates for each PIN validation algorithm if it is selected or not. Properties are read only.	Extends: **JxfsType**
Class	**JxfsPINChipPresentation Modes**	PIN chip presentation algorithms selector class. Indicates which presentation algorithms for chip PIN validation are supported.	Extends: **JxfsType**
Class	**JxfsPINValidationData**	Abstract data class. Root of a hierarchy of data objects that contain data for PIN verification and used in	Extends: **JxfsType**

		validationPIN() method.	
Class	**JxfsPINValidationDataForDES**	Data class for PIN verification using DES algorithm.	Extends: **JxfsPINValidationData**
Class	**JxfsPINValidationDataForEC**	Data class for PIN verification using EUROCHEQUE specification.	Extends: **JxfsPINValidationData**
Class	**JxfsPINValidationDataForVISA**	Data class for PIN verification using VISA specification.	Extends: **JxfsPINValidationData**
Class	**JxfsPINOffsetData**	Data class for creating a PIN offset.	Extends: **JxfsPINValidationData**
Class	**JxfsPINBlockData**	Data class for creating a PIN block.	Extends: **JxfsPINValidationData**
Class	**JxfsPINChipValidationData**	Abstract data class for all PIN chip validation modes.	Extends: **JxfsType**
Class	**JxfsPINChipValidationDataClear**	Data class for PIN chip validation mode Clear. Used as parameter in *validatePINChip* () method.	Extends: **JxfsPINChipValidationData**
Class	**JxfsPINValidationResult**	Data class that contains the result of a PIN validation operation.	Extends: **JxfsType**
Class	**JxfsPINOffset**	Data class that contains computed PIN offset.	Extends: **JxfsType**
Class	**JxfsPINBlock**	Data class that contains computed PIN block.	Extends: **JxfsType**
Class	**JxfsPINChipValidationResult**	Data class that contains the result of a PINchip validation operation.	Extends: **JxfsType**
Class	**JxfsPINCryptoModes**	Encyption modes selector class. Indicates for each encryption mode if it is selected or not. Properties are read only.	Extends: **JxfsType**
Class	**JxfsPINEMVCryptoModes**	Encryption modes selector class. Indicates for each encryption mode if it is selected or not. Properties are read only. it is only for EMV RSA keys	Extends: **JxfsType**
Class	**JxfsPINKeyDetail**	Data class containing information about a key from the device's key table.	Extends: **JxfsType**
Class	**JxfsPINEMVRSAKeyToImport**	Data class containing input data for *importEMVRSAPublicKey()* method, specially for RSA keys	Extends: **JxfsType**
Class	**JxfsPINKeyToImport**	Data class containing input data for *importKey()* method	Extends: **JxfsType**
Class	**JxfsPINInitialization**	Data class that contains result data from initialization of security module.	Extends: **JxfsType**
Class	**JxfsPINKeyVerificationData**	Data class that contains result data from an import key operation.	Extends: **JxfsType**
Class	**JxfsPINCryptoData**	Data class that contains input	Extends:

		data for encrypt/decrypt operations.	**JxfsType**
Class	**JxfsPINMACData**	Data class that contains input data for MAC generation operation.	Extends: **JxfsPINCryptoData**
Class	**JxfsPINCryptoResult**	Data class that contains result data from cryptographic operations.	Extends: **JxfsType**
Class	**JxfsPINKeyUses**	Data class that contains information on allowed uses for a key.	Extends: **JxfsType**
Class	**JxfsPINIdKeyModes**	Data class that contains information on implemented uses of ID key.	Extends: **JxfsType**
Class	**JxfsPINEMVRSAIntegrit y**	Data class that contains information about the type of verification to compute for the RSA key to import.	Extends: **JxfsType**
Class	**JxfsPINIdKeyModes**	Data class that contains iformation to cimpute a SHA 1 digest or the result of the computation.	Extends: **JxfsType**
Class	**JxfsPINImportRSAPublic Key**	Data class that contains class contains data required as input for *importRSAPublicKey()* operation.	Extends: **JxfsType**
Class	**JxfsPINExportedRSAPub licKey**	data returned on *JxfsOperationCompleteEvent* of the *exportRSAPublicKey()* operation	Extends: **JxfsType**
Class	**JxfsPINExportRSAPublic Key**	Data class that contains information data that specifies the RSA public key to export.	Extends: **JxfsType**
Class	**JxfsPINImportRSADESE ncipheredPublicKey**	Data class that class contains data required as input for *importRSADESEncipheredP ublicKey()* operation.	Extends: **JxfsType**
Class	**JxfsPINGenerateRSAKey Pair**	Data class that class contains data required as input for *generateRSAKeyPair()* operation.	Extends: **JxfsType**
Class	**JxfsPINExportId**	Data class that class contains data retrieved by the PIN device and which uniquely identifies the PIN device.	Extends: **JxfsType**
Class	**JxfsPINExportCertificate**	Data class that class contains data required as output for *exportCertificate()* operation.	Extends: **JxfsType**
Class	**JxfsPINCertificateType**	Data class that class contains the type, primary or secondary, of certificate exported from the encryptor.	Extends: **JxfsType**
Class	**JxfsPINCertificateKeyTy pe**	Data class that class contains data required as input for *exportCertificate()* operation.	Extends: **JxfsType**
Class	**JxfsPINRSAHashAlgorith ms**	This class provides properties and methods to query which type of hash algorithms is to be processed.	Extends: **JxfsType**

Class	**JxfsPINRSASignatureAlgo**	This class provides properties and methods to query which type of RSA Signature algorithms is to be processed.	Extends: **JxfsType**
Class	**JxfsPINRSAExponent**	This class provides properties and methods to query which exponent value of the RSA key pair to be generated.	Extends: **JxfsType**
Class	**JxfsPINRSAKeyVerificationData**	This class contains information about the imported RSA Public key.	Extends: **JxfsType**
Class	**JxfsPINRSADESkeyVerificationData**	This class contains information about the imported RSA DES enciphered public key.	Extends: **JxfsType**
Class	**JxfsPINRSADESLength**	This class specifies the key length that was loaded.	Extends: **JxfsType**
Class	**JxfsPINRSADESCheckMode**	This class specifies the mode that was used to create the check value.	Extends: **JxfsType**
Class	**JxfsPINRSAKeyType**	This class specifies the private signature to use.	Extends: **JxfsType**
Class	**JxfsPINRemoteKeyLoadModes**	This class provides properties and methods to query which remote key loading modes are supported by a secure PIN device service.	Extends: **JxfsType**
Class	**JxfsPINRSAAlgorithm**	This class provides properties and methods to query which RSA algorithm are supported by the secure PIN device service.	Extends: **JxfsType**
Class	**JxfsEvent**	Abstract class from which all Jxfs event classes are extended	Extends: **java.util. EventObject**
Class	**StatusEvent OperationCompleteEvent IntermediateEvent**	The Device Service creates *Event* event instances of this class and delivers them through the J/XFS PIN Device Control's event callbacks to the application	Extends: **JxfsEvent**
Class	**JxfsException**	Exception class. The J/XFS PIN Device Control creates and throws exceptions on method failure and property access failure.	Extends: **java.lang.Exception**

3 Device behavior

3.1 Device open()

During the device open call the Device Service tries to access the connected device. This fails for the following circumstances:

JXFS_E_HARDWAREERROR	If the device could not be accessed. This may be that the device is not connected or broken. This is returned as the result property in an OperationCompleteEvent.
JXFS_E_OPEN	The open was already done by this Device Control. This is returned as the errorCode field in a JxfsException.

3.2 Handling of null parameters

If null is passed as a method parameter, a JxfsException exception with the error Code property set to JXFS_E_PARAMETER_INVALID will be thrown, unless the handling of a null parameter is explicitly specified for a particular method.

4 Classes and Interfaces

All operation methods return an identificationID. If an operation cannot be processed because of an error detected before the asynchronous processing of the method begins (i.e. before the calling thread returns) a JxfsException is thrown.
After processing has taken place, an OperationCompleteEvent is generated which contains detailed information about the status of the operation, i.e., if it failed or succeeded, and eventually additional data as a result.

The Constants, Error Codes, Exceptions, Status Codes and Support Classes that are used in the methods are described in special chapters at the end of the documentation.

4.1 Access to properties

Please note the following when determining the meaning of a property's **Access**:

R	The property is read only.
W	The property is write only.
R/W	The property may be read or written.

To access these properties the applications must use the appropriated methods specified by the JavaBean specification. Note that boolean properties are read using *isProperty* method instead of *getProperty*.

getProperty

Syntax	**Propert*y get*Property *() throws JxfsException***
Description	Returns the requested property.
Parameter	**None**
Event	No additional events are generated.
Exceptions	Some possible JxfsException *value codes*. Common values are: JXFS_E_CLOSED JXFS_E_UNREGISTERED JXFS_E_REMOTE

set*Property*

Syntax	**void *set*Property *(value) throws JxfsException***
Description	Sets the requested property.
Parameter	The desired property value.
Event	No additional events are generated
Exceptions	Some possible JxfsException *value codes*. Common values are: JXFS_E_CLOSED JXFS_E_UNREGISTERED JXFS_E_REMOTE JXFS_E_PARAMETER_INVALID

4.2 Exceptions

All the methods described for the specified interfaces can throw at least some of the following exceptions:

Value	**Meaning**
JXFS_E_CLOSED	The Device Control has not been opened.
JXFS_E_UNREGISTERED	The device is not registered at the JxfsDeviceManager.
JXFS_E_REMOTE	A network error occurred.
JXFS_E_PARAMETER_INVALID	A parameter is invalid.
JXFS_E_NOT_SUPPORTED	The function is not supported.

Only if a method can throw additional exceptions this is explicitly mentioned.

4.3 IJxfsPINKeypadControl

4.3.1 Introduction

The J/XFS PIN Keypad Device Control Subclass is defined in JxfsPINKeypad and is a subclass of JxfsBaseControl. Its interface is defined in IJxfsPINKeypadControl interface which is a subclass of IJxfsBaseControl interface. The purpose of the J/XFS PIN Keypad Device Control object is to allow passing data and control between the application and the device support code so that the associated device can be accessed.

The JxfsPINKeypad class represents a physical PIN Keypad device with basic input keypad functions. There are no built-in security functions.

Summary

Although IJxfsPINKeypadControl is an interface, and therefore properties do not apply, properties are detailed here with the objective to provide guidance on the implementation of those classes that will implement this interface.

Therefore, the IJxfsPINKeypadControl consists on the following methods:
- Getters of listed properties.
- Methods listed.

Implements : **Extends : IJxfsBaseControl**

Property	Type	Access	Initialized after
supportedFDKeys	java.util.Vector	R	After successful open
supportedFKeys	**JxfsPINFKeySet**	R	After successful open
inputRawSupported	boolean	R	After successful open
inputCookedSupported	boolean	R	After successful open
beepOnPressSupported	boolean	R	After successful open
eventOnStartSupported	boolean	R	After successful open

Method	Return	May use after
get*Property*	*Property*	After successful open
readData	identificationID	After successful open

4.3.2 Properties

supportedFDKeys Property (R)

Type	***java.util.Vector***
Initial Value	Depends on device.
Description	This vector contains a list of all function descriptor keys (FDKeys) supported by the device. Each vector element is a **JxfsPINFDKey** object that contains its key code and position information. See JxfsPINFDKey class description for more information. If empty, then no FDKeys are supported.

supportedFKeys Property (R)

Type	***JxfsPINFKeySet***
Initial Value	Null until open.
Description	Indicates the set of function keys supported by the device.

inputRawSupported (R)

Type	***boolean***
Initial Value	Depends on device.
Description	Specifies if raw input mode is supported by the device, where each key pressed during an input operation will generate an intermediate event. These events will contain information about pressed keys.

Value	**Meaning**
FALSE	Raw input mode is not supported.
TRUE	Raw input mode is supported.

inputCookedSupported (R)

Type	***boolean***
Initial Value	Depends on device.
Description	Specifies if cooked input mode is supported by the device, where no intermediate events per key pressed are generated. Data entered during an input operation is provided in the *OperationCompleteEvent* event.

Value	**Meaning**
FALSE	Cooked input mode is not supported.
TRUE	Cooked input mode is supported.

beepOnPressSupported (R)

Type	***boolean***
Initial Value	Depends on device.
Description	Specifies if the device has controllable capability of emitting an audible sound when a key is pressed.

Value	**Meaning**
FALSE	Device has no controllable beep capability.
TRUE	Device has controllable beep capability.

eventOnStartSupported (R)

Type	***boolean***
Initial Value	Depends on service.
Description	Specifies if the service has the capability to send the intermediate event JXFS_I_PIN_READ_STARTED

Value	**Meaning**
FALSE	The service does not have this capability.
TRUE	The service has this capability.

4.3.3 Methods

readData Method

Syntax	***identificationID readData (JxfsPINReadMode readMode) throws JxfsException;***
Description	This command activates the PIN Keypad to read a data entry. Digits are read until the value of *maxLength* property of *readMode* parameter is reached (if *autoEnd* property of *readMode* is set to TRUE), or a termination key is pressed. If *maxLength* is set to zero and no termination keys are specified, operation will not terminate until cancelled. Each key pressed is notified as an intermediate event if *inputMode* property of *readMode* parameter is set to JXFS_PIN_INPUT_RAW. If *inputMode* is set to JXFS_PIN_INPUT_COOKED, then, a single *OperationCompleteEvent* event (containing input data) is issued when input operation terminates.
Parameter	see below

Type	IO	Name	Meaning
JxfsPINReadMode	I	readMode	A data object that contains all the data required to perform a data entry (see *JxfsPINReadMode* class specification).

Event

OperationCompleteEvent
When an input operation is completed an *OperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered OperationCompleteListeners

Field	Value
operationID	JXFS_O_PIN_READPIN
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully.
	JXFS_E_CANCELLED Operation was cancelled.
	JXFS_E_PIN_READ_FAILURE Read error.
data	A **JxfsPINReadData** object.

IntermediateEvent
Every key pressed generates an intermediate event if *inputMode* property is set to JXFS_PIN_INPUT_RAW.
IntermediateEvent events are sent by PIN Device Control to all registered IntermediateListeners

Field	Value
operationID	JXFS_O_PIN_READPIN
identificationID	Identification Id of operation.
reason:	
	JXFS_I_PIN_KEY_PRESSED A key has been pressed.
data	A **JxfsPINPressedKey** object.

Exceptions

Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	Meaning
JXFS_E_PIN_KEYINVALID	At least one of the specified active function keys or FDKeys is invalid.
JXFS_E_PIN_NOACTIVEKEYS	No active function key or FDKey specified.

JXFS_E_PIN_KEYNOTSUPPORTED	At least one of the specified active function keys or FDKeys (*activeFKeys* or *activeFDKeys* properties of *readMode* parameter) is not supported by the device service.
JXFS_E_PIN_MINIMUNLENGTH	The *minLength* property is invalid or greater than the *maxLength* property.

readData Method

Syntax ***identificationID readData (JxfsPINReadMode2 readMode) throws JxfsException;***

Description This command activates the PIN Keypad to read a data entry.

Digits are read until the value of *maxLength* property of *readMode* parameter is reached (if *autoEnd* property of *readMode* is set to TRUE), or a termination key is pressed. If *maxLength* is set to zero and no termination keys are specified, operation will not terminate until cancelled.

Each key pressed is notified as an intermediate event if *inputMode* property of *readMode* parameter is set to JXFS_PIN_INPUT_RAW. If *inputMode* is set to JXFS_PIN_INPUT_COOKED, then, a single *OperationCompleteEvent* event (containing input data) is issued when input operation terminates.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINReadMode2	I	readMode	A data object that contains all the data required to perform a data entry (see *JxfsPINReadMode* class specification).

Event

OperationCompleteEvent
When an input operation is completed an *OperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered OperationCompleteListeners

Field	**Value**
operationID	JXFS_O_PIN_READPIN
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully.
	JXFS_E_CANCELLED Operation was cancelled.
	JXFS_E_PIN_READ_FAILURE Read error.
data	A **JxfsPINReadData** object.

IntermediateEvent
Every key pressed generates an intermediate event if *inputMode* property is set to JXFS_PIN_INPUT_RAW.
IntermediateEvent events are sent by PIN Device Control to all registered IntermediateListeners

Field	**Value**
operationID	JXFS_O_PIN_READPIN
identificationID	Identification Id of operation.
reason:	
	JXFS_I_PIN_KEY_PRESSED A key has been pressed.
data	A **JxfsPINPressedKey** object.

IntermediateEvent

If the eventOnStart property is set, the service sends this event when the operation is really started. That is the moment when the device begins accepting data entered by the user.

Field	**Value**
operationID	JXFS_O_PIN_READPIN
identificationID	Identification Id of operation.
reason:	
	JXFS_I_PIN_READ_STARTED The device is ready for input operation.
data	null

Exceptions

Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PIN_KEYINVALID	At least one of the specified active function keys or FDKeys is invalid.
JXFS_E_PIN_NOACTIVEKEYS	No active function key or FDKey specified.
JXFS_E_PIN_KEYNOTSUPPORTED	At least one of the specified active function keys or FDKeys (*activeFKeys* or *activeFDKeys* properties of *readMode* parameter) is not supported by the device service.
JXFS_E_PIN_MINIMUNLENGTH	The *minLength* property is invalid or greater than the *maxLength* property.

4.4 IJxfsSecurePINKeypadControl

4.4.1 Introduction

The J/XFS Secure PIN Keypad Device Control Subclass is defined in JXFSecurePINKeypad and is a subclass of JxfsPINKeypad. The Secure PIN Keypad Device Control is intended to match physical PIN Keypad devices with the following extended security capabilities:

- PIN secure read,
- PIN verification and
- Cryptographic services.

Its interface is defined in IJxfsSecurePINKeypadControl interface which is a subclass of IJxfsPINKeypadControl interface.

Summary

Although IJxfsSecurePINKeypadControl is an interface, and therefore properties do not apply, properties are detailed here with the objective to provide guidance on the implementation of those classes that will implement this interface.

Therefore, the IJxfsSecurePINKeypadControl consists on the following methods:

- Getters of listed properties.
- Methods listed.

Implements : **Extends :** *IJxfsPINKeypadControl*

Property	Type	Access	Initialized after
supportedPINFormats	**JxfsPINFormats**	R	After successful open
supportedValidationAlgorithms	**JxfsPINValidationAlgorithms**	R	After successful open
supportedChipPresentationModes	**JxfsPINChipPresentationModes**	R	After successful open

Method	Return	May use after
get*Property*	*Property*	After successful open
secureReadPIN	identificationID	After successful open
createOffset	identificationID	After successful open
createPINBlock	identificationID	After successful open
validatePIN	identificationID	After successful open
createOffsetSecure	identificationID	After successful open
createPINBlockSecure	identificationID	After successful open
validatePINSecure	identificationID	After successful open
validatePINChip	identificationID	After successful open

4.4.2 Properties

supportedPINFormats Property (R)

Type	***JxfsPINFormats***
Initial Value	Null until open.
Description	Specifies the supported PIN formats.

supportedValidationAlgorithms Property (R)

Type	***JxfsPINValidationAlgorithms***
Initial Value	Null until open.
Description	Specifies the supported algorithms for PIN validation.

supportedChipPresentationModes Property (R)

Type	***JxfsPINChipPresentationModes***
Initial Value	Depends on device.
Description	Specifies the supported presentation algorithms for chip PIN validation.

4.4.3 Methods

secureReadPIN Method

Syntax

identificationID secureReadPIN (JxfsPINReadMode readMode) throws JxfsException;

Description

This command activates the PIN Keypad to read a PIN entry in a secure way.
Entered data is not passed to the application but retained for further cryptographic operation (like PIN validation, PIN offset generation or PIN Block generation).

Digits are read until the value of *maxLength* property of *readMode* parameter is reached (if *autoEnd* property of *readMode* is set to TRUE), or a termination key is pressed. If *maxLength* is set to zero and no termination keys are specified, operation will not terminate until cancelled.

Each key pressed is notified as an intermediate event if *inputMode* property of *readMode* parameter is set to JXFS_PIN_INPUT_RAW. If *inputMode* is set to JXFS_PIN_INPUT_COOKED, then, a single *OperationCompleteEvent* event (containing input data) is issued when input operation terminates.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINReadMode	I	readMode	A data object that contains all the data required to perform a data entry (see *JxfsPINReadMode* class specification).

Event

OperationCompleteEvent
When an input operation is completed an *OperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered OperationCompleteListeners

Field	**Value**
operationID	JXFS_O_PIN_READPIN
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully.
	JXFS_E_CANCELLED Operation was cancelled.
	JXFS_E_PIN_READ_FAILURE Read error.
data	A **JxfsPINReadData** object.

IntermediateEvent
Every key pressed generates an intermediate event if *inputMode* property is set to JXFS_PIN_INPUT_RAW.
IntermediateEvent events are sent by PIN Device Control to all registered IntermediateListeners

Field	**Value**
operationID	JXFS_O_PIN_READPIN
identificationID	Identification Id of operation.
reason:	
	JXFS_I_PIN_KEY_PRESSED A key has been pressed.
data	A **JxfsPINPressedKey** object.

Exceptions

Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**

JXFS_E_PIN_KEYINVALID	At least one of the specified active function keys or FDKeys is invalid.
JXFS_E_PIN_NOACTIVEKEYS	No active function key or FDKey specified.
JXFS_E_PIN_KEYNOTSUPPORTED	At least one of the specified active function keys or FDKeys (*activeFKeys* or *activeFDKeys* properties of *readMode* parameter) is not supported by the device service.
JXFS_E_PIN_MINIMUNLENGTH	The *minLength* property is invalid or greater than the *maxLength* property.

secureReadPIN Method

Syntax

identificationID secureReadPIN (JxfsPINReadMode2 readMode) throws JxfsException;

Description

This command activates the PIN Keypad to read a PIN entry in a secure way.
Entered data is not passed to the application but retained for further cryptographic operation (like PIN validation, PIN offset generation or PIN Block generation).

Digits are read until the value of *maxLength* property of *readMode* parameter is reached (if *autoEnd* property of *readMode* is set to TRUE), or a termination key is pressed. If *maxLength* is set to zero and no termination keys are specified, operation will not terminate until cancelled.

Each key pressed is notified as an intermediate event if *inputMode* property of *readMode* parameter is set to JXFS_PIN_INPUT_RAW. If *inputMode* is set to JXFS_PIN_INPUT_COOKED, then, a single *OperationCompleteEvent* event (containing input data) is issued when input operation terminates.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINReadMode	I	readMode	A data object that contains all the data required to perform a data entry (see *JxfsPINReadMode* class specification).

Event

OperationCompleteEvent
When an input operation is completed an *OperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered OperationCompleteListeners

Field	**Value**
operationID	JXFS_O_PIN_READPIN
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_READ_FAILURE Read error.
data	A **JxfsPINReadData** object.

IntermediateEvent
Every key pressed generates an intermediate event if *inputMode* property is set to JXFS_PIN_INPUT_RAW.

IntermediateEvent events are sent by PIN Device Control to all registered IntermediateListeners

Field	**Value**
operationID	JXFS_O_PIN_READPIN
identificationID	Identification Id of operation.
reason:	
	JXFS_I_PIN_KEY_PRESSED A key has been pressed.
data	A **JxfsPINPressedKey** object.

IntermediateEvent
If the eventOnStart property is set, the service sends this event when the operation is really started. That is the moment when the device begins accepting data entered by the user.

Field	**Value**
operationID	JXFS_O_PIN_READPIN
identificationID	Identification Id of operation.
reason:	
	JXFS_I_PIN_READ_STARTED The device is ready for input operation.
data	null

Exceptions

Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PIN_KEYINVALID	At least one of the specified active function keys or FDKeys is invalid.
JXFS_E_PIN_NOACTIVEKEYS	No active function key or FDKey specified.
JXFS_E_PIN_KEYNOTSUPPORTED	At least one of the specified active function keys or FDKeys (*activeFKeys* or *activeFDKeys* properties of *readMode* parameter) is not supported by the device service.
JXFS_E_PIN_MINIMUNLENGTH	The *minLength* property is invalid or greater than the *maxLength* property.

createOffset Method

Syntax

identificationID createOffset (JxfsPINOffsetData offsetData) throws JxfsException;

Description

This function is used to generate a PIN Offset that is used to verify PINs using the *validatePIN()* method with DES validation algorithm.

The PIN offset is computed by combining validation data with the keypad entered PIN.

This method clears the PIN.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINOffsetData	I	offsetData	A data object that contains all the data required to create the PIN offset (see *JxfsPINOffsetData* class specification).

Event

OperationCompleteEvent
When the operation completes an *OperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered OperationCompleteListeners. In addition a data object is returned:

Field	**Value**
operationID	JXFS_O_PIN_CREATEOFFSET

identificationID	Identification Id of complete operation.
result	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_NO_PIN PIN has not been entered or has been cleared. JXFS_E_PIN NOT_ALLOWED PIN entered by the user is not allowed. JXFS_E_PIN_KEY_NOT_FOUND The specified key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by this key. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	A **JxfsPINOffset** object. It contains the computed PIN offset

Exceptions

Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PIN_NOTSUPPORTEDCAP	Offset generation is not supported.

createPINBlock Method

Syntax

identificationID createPINBlock (JxfsPINBlockData pinBlockData) throws JxfsException;

Description

This method takes the account information and a PIN entered by the user to build a formatted PIN. Encrypting this formatted PIN once or twice returns a PIN block which can be written on a magnetic card or sent to a host.

The PIN block can be calculated using one of the formats specified in the *supportedPINFormats* property.

The PIN block is computed by combining customer data with the keypad entered PIN.

This command clears the PIN.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINBlockData	I	pinBlockData	A data object that contains all the data required to create the PIN block (see *JxfsPINBlockData* class specification).

Event

OperationCompleteEvent

When the operation completes an *OperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered OperationCompleteListeners.

Field	**Value**
operationID	JXFS_O_PIN_CREATEPINBLOCK
identificationID	Identification Id of complete operation.
result	JXFS_RC_SUCCESSFUL Operation completed successfully.

		JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_NO_PIN PIN has not been entered or has been cleared. JXFS_E_PIN_NOT_ALLOWED PIN entered by the user is not allowed. JXFS_E_PIN_KEY_NOT_FOUND The specified key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by this key. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
	data	A **JxfsPINBlock** object. It contains the computed PIN block.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PIN_FORMAT_NOTSUPPORTED	The specified PIN block format is not supported.

validatePIN Method

Syntax ***identificationID validatePIN (JxfsPINValidationData validationData) throws JxfsException;***

Description The previously entered PIN is combined with the requisite data specified by the PIN validation algorithm and locally verified for correctness.

The validationData object should specify the validation algorithm to be used for PIN validation as well as all needed data to perform the validation (*see JxfsPINValidationData* class specification)

This method clears the PIN.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINValidationData	I	validationData	Validation data object containing specific data for the actual PIN validation algorithm to be used *(see JxfsPINValidationData* class specification).

Event **OperationCompleteEvent**

When the operation completes an *OperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered OperationCompleteListeners.

Field	**Value**
operationID	JXFS_O_PIN_VALIDATEPIN
identificationID	Identification Id of complete operation.
result	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled.

	JXFS_E_PIN_NO_PIN PIN has not been entered or has been cleared. JXFS_E_PIN_NOT_ALLOWED PIN entered by the user is not allowed. JXFS_E_PIN_KEY_NOT_FOUND The specified key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by this key. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	A **JxfsPINValidationResult** object. It contains the results of the validation.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PIN_NOTSUPPORTEDCAP	The requested validation algorithm is not supported.

createOffsetSecure Method

Syntax ***identificationID createOffsetSecure (JxfsPINOffsetData offsetData) throws JxfsException;***

Description This function is used to generate a PIN Offset that is used to verify PINs using the *validatePIN()* method with DES validation algorithm.

With combined MSD-PIN devices, this function does not require that validation data be first read from the card with the MSD component and then returned to the device as a parameter. Instead, the validation data is automatically read from the card in the device.
The behavior is as follows:
1 – If card is present in reader and ejectCurrent property is false then go to 5.
2 – If card is present in reader and ejectCurrent property is true then eject the card.
3 – Arm the device to accept a magnetic stripe card.
4 – Poll card status and verify that card is seated.
5 – Perform the intended function using the offset data read from the card.
6 – Eject the card if ejectWhenComplete property is true.

This method clears the PIN.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINOffsetData	I	offsetData	A data object that contains all the data required to create the PIN offset (see *JxfsOffsetData* class specification).

Event **OperationCompleteEvent**
When the operation completes an *OperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered OperationCompleteListeners. In addition a data object is returned:

Field	**Value**
operationID	JXFS_O_PIN_CREATEOFFSET_SECURE
identificationID	Identification Id of complete operation.

result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_NO_PIN PIN has not been entered or has been cleared. JXFS_E_PIN NOT_ALLOWED PIN entered by the user is not allowed. JXFS_E_PIN_KEY_NOT_FOUND The specified key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by this key. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason. JXFS_E_MSD_READFAILURE No read conditions were satisfied JXFS_E_MSD_NOMEDIA Media was removed before operation completion. JXFS_E_MSD_INVALIDMEDIA No appropriated media was found. JXFS_E_MSD_MEDIAJAMMED Media is jammed. JXFS_E_MSD_SHUTTERFAIL Shutter could not be opened.
data	A **JxfsPINOffset** object. It contains the computed PIN offset

Exceptions

Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PIN_NOTSUPPORTEDCAP	Secure offset generation is not supported.
JXFS_E_MSD_NOTSUPPORTEDTRACK	Track specified in *validationTrackNumber* property is not supported by the device.

createPINBlockSecure Method

Syntax

identificationID createPINBlockSecure (JxfsPINBlockData pinBlockData) throws JxfsException;

Description

This method takes the account information and a PIN entered by the user to build a formatted PIN. Encrypting this formatted PIN once or twice returns a PIN block which can be written on a magnetic card or sent to a host.

The PIN block can be calculated using one of the formats specified in the *supportedPINFormats* property.

The PIN block is computed by combining customer data with the keypad entered PIN.

With combined MSD-PIN devices, this function does not require that customer data be returned to the device as a parameter. Instead, the customer data is automatically read from the card in the device.
The behavior is as follows:

1 – If card is present in reader and ejectCurrent property is false then go to 5.
2 – If card is present in reader and ejectCurrent property is true then eject the card.
3 – Arm the device to accept a magnetic stripe card.
4 – Poll card status and verify that card is seated.
5 – Perform the intended function using the customer data read from the card.
6 – Eject the card if ejectWhenComplete property is true.

This command clears the PIN.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINBlockData	I	pinBlockData	A data object that contains all the data required to create the PIN block (see *JxfsPINBlockData* class specification).

Event

OperationCompleteEvent
When the operation completes an *OperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered OperationCompleteListeners.

Field	**Value**
operationID	JXFS_O_PIN_CREATEPINBLOCK_SECURE
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_NO_PIN PIN has not been entered or has been cleared. JXFS_E_PIN_NOT_ALLOWED PIN entered by the user is not allowed. JXFS_E_PIN_KEY_NOT_FOUND The specified key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by this key. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason. JXFS_E_MSD_READFAILURE No read conditions were satisfied JXFS_E_MSD_NOMEDIA Media was removed before operation completion. JXFS_E_MSD_INVALIDMEDIA No appropriated media was found. JXFS_E_MSD_MEDIAJAMMED Media is jammed. JXFS_E_MSD_SHUTTERFAIL Shutter could not be opened.
data	A **JxfsPINBlock** object. It contains the computed PIN block.

Exceptions

Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	Meaning
JXFS_E_PIN_NOTSUPPORTEDCAP	Secure block generation is not supported.
JXFS_E_PIN_FORMAT_NOTSUPPORTED	The specified PIN block format is not supported.
JXFS_E_MSD_NOTSUPPORTEDTRACK	Track specified in *validationTrackNumber* property is not supported by the device.

validatePINSecure Method

Syntax

identificationID validatePINSecure (JxfsPINValidationData validationData) throws JxfsException;

Description

The previously entered PIN is combined with the requisite data specified by the DES validation algorithm and locally verified for correctness.

With combined MSD-PIN devices, this function does not require that offset and/or validation data be returned to the device as parameters. Instead, offset and/or validation data can be automatically read from the card in the device.
The behavior is as follows:
1 – If card is present in reader and ejectCurrent property is false then go to 5.
2 – If card is present in reader and ejectCurrent property is true then eject the card.
3 – Arm the device to accept a magnetic stripe card.
4 – Poll card status and verify that card is seated.
5 – Perform the intended function using the data read from the card.
6 – Eject the card if ejectWhenComplete property is true.

This method clears the PIN.

Parameter

Type	IO	Name	Meaning
JxfsPINValidationData	I	validationData	Validation data object containing specific data for the actual PIN validation algorithm to be used *(see JxfsPINValidationData* class specification).

Event

OperationCompleteEvent
When the operation completes an *OperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered OperationCompleteListeners.

Field	Value
operationID	JXFS_O_PIN_VALIDATEPIN_SECURE
identificationID	Identification Id of complete operation.
result	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_NO_PIN PIN has not been entered or has been cleared. JXFS_E_PIN_NOT_ALLOWED PIN entered by the user is not allowed.

	JXFS_E_PIN_KEY_NOT_FOUND The specified key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by this key. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason. JXFS_E_MSD_READFAILURE No read conditions were satisfied JXFS_E_MSD_NOMEDIA Media was removed before operation completion. JXFS_E_MSD_INVALIDMEDIA No appropriated media was found. JXFS_E_MSD_MEDIAJAMMED Media is jammed. JXFS_E_MSD_SHUTTERFAIL Shutter could not be opened.
data	A **JxfsPINValidationResult** object. It contains the results of the validation.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PIN_NOTSUPPORTEDCAP	Secure PIN validation is not supported.
JXFS_E_MSD_NOTSUPPORTEDTRACK	Tracks specified in *validationTrackNumber* and/or *offsetTrackNumber* properties are not supported by the device.

validatePINChip Method

Syntax ***identificationID validatePINChip (java.lang.String aCCDeviceName, JxfsPINChipValidationData validationData) throws JxfsException;***

Description The previously entered PIN is combined with the requisite data specified by the chip PIN presentation algorithm and presented to the chip card device for correctness verification.

The validationData object specifies all the needed data to perform the validation (*see JxfsPINChipValidationData* class and subclasses specifications)

This method clears the PIN.

Parameter	Type	IO	Name	Meaning
	java.lang.String	I	aCCDeviceName	The name of the Chip Card device to be used for PIN validation. It is the responsibility of the application to ensure the chip card has already been inserted. It is the responsibility of the J/XFS device service to instantiate a J/XFS Chip Card Control and to use it exclusively to access the chip card. If the chip card device is already claimed by someone else, a JXFS_E_CLAIMED exception is thrown. The device service must release ownership of the device after using it. During the validation of the PIN the application must not access the chip card; only the PinPad device service has the right to access the chip card.
	JxfsPINChipValidationData	I	validationData	Validation data object containing specific data for the actual PIN validation algorithm to be used *(see JxfsPINChipValidationData* class specification).

Event **OperationCompleteEvent**

When the operation completes an *OperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered OperationCompleteEvent listeners.

Field	Value
operationID	JXFS_O_PIN_VALIDATEPINCHIP
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully.
	JXFS_E_CANCELLED Operation was cancelled.
	JXFS_E_PIN_NO_PIN PIN has not been entered or has been cleared.
	JXFS_E_PIN_NOT_ALLOWED PIN entered by the user is not allowed.
	JXFS_E_CCD_IOERROR IO error occurred. No data is available. Verification could not be performed.
	JXFS_E_CCD_NOMEDIA Media was removed before operation completion
	JXFS_E_CCD_INVALIDMEDIA No appropriated media was found.
	JXFS_E_CCD_MEDIAJAMMED Media is jammed.
	JXFS_E_CCD_SHUTTERFAIL Shutter could not be opened.

	JXFS_E_CCD_BADDATA Chip reported data was bad. JXFS_E_CCD_BADPROTOCOL Protocol not supported.
data	A **JxfsPINChipValidationResult** object. It contains the results of the validation.

IntermediateEvent

IntermediateEvent events can be sent by PIN Device Control to all registered IntermediateListeners

Field	**Value**
operationID	JXFS_O_CCD_CHIPIO
identificationID	Identification Id of operation.
reason:	
	JXFS_I_CCD_NO_MEDIA_PRESENT The read operation request cannot progress because there is no media inserted. JXFS_I_CCD_MEDIA_INSERTED The read operation request continues because a media has been inserted.
data	Null

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PIN_CHIPPRES_ NOTSUPPORTED	The requested chip presentation algorithm is not supported.

4.5 IJxfsCrypto

4.5.1 Introduction

The cryptographic services interface provides generic cryptography functions. It handles a key table and allows the user to encrypt, decrypt or calculate check codes using keys from its table. This interface is used for the sake of clarity; to separate the generic cryptographic functions from the PIN related cryptographic functions. The JxfsSecurePINKeypad class implements this interface.

Summary

Implements : - **Extends :** *IJxfsPINKeypadControl*

Property	Type	Access	Initialized after
supportedCryptoModes	**JxfsPINCryptoModes**	R	After successful open
numberOfKeys	int	R	After successful open
idKey	**JxfsPINIdKeyModes**	R	After successful open

Method	Return	May use after
get*Property*	*Property*	After successful open
decrypt	identificationID	After successful open
encrypt	identificationID	After successful open
generateMAC	identificationID	After successful open
getKeyInfo	**JxfsPINKeyDetail**	After successful open
getKeyNameList	java.util.Vector	After successful open
importKey	identificationID	After successful open
initialize	identificationID	After successful open
importEMVRSAPublicKey	identificationID	After successful open
computeSHA1Digest	identificationID	After successful open
deleteKey	identificationID	After successful open
importRSAPublicKey	identificationID	After successful open
exportRSAPublicKey	identificationID	After successful open
importRSADESEnciphered PublicKey	identificationID	After successful open
exportRSADESEnciphered PublicKey	identificationID	After successful open
generateRSAKeyPair	identificationID	After successful open
exportPINId	identificationID	After successful open
importCertificate	identificationID	After successful open
exportCertificate	identificationID	After successful open
replaceCertificate	identificationID	After successful open
startKeyExchange	identificationID	After successful open

4.5.2 Properties

supportedCryptModes Property (R)

Type	***JxfsPINCryptoModes***
Initial Value	Depends on device.
Description	Specifies the supported encryption modes.

numberOfKeys Property (R)

Type	***int***
Initial Value	Depends on device.
Description	Specifies the number of keys that may be stored by the device.

idKey Property (R)

Type	***JxfsPINIdKeyModes***
Initial Value	Depends on device.
Description	Specifies whether an ID key is supported or not.

4.5.3 Methods

decrypt Method

Syntax	***identificationID decrypt (JxfsPINCryptoData decryptData) throws JxfsException;***
Description	Deciphers data with the currently selected algorithm and the specified key name.

Parameter

Type	IO	Name	Meaning
JxfsPINCryptoData	I	decryptData	Contains the data and additional information required to perform a decrypt operation. See *JxfsPINCryptoData* specification).

Event

OperationCompleteEvent
When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	Value
operationID	JXFS_O_PIN_DECRYPT
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_KEY_NOT_FOUND The specified key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by this key JXFS_E_PIN_LENGTH_ERROR The length of the start value specified is not supported. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	A **JxfsPINCryptoResult** object. It contains the results of the decryption.

Exceptions

Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	Meaning
JXFS_E_PIN_CRYPTNOTSUPPORTED	The decryption method is not supported.

encrypt Method

Syntax	***identificationID encrypt (JxfsPINCryptoData encryptData) throws JxfsException;***
Description	Encrypts data with the currently selected algorithm and the specified key name.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINCryptoData	I	encryptData	Contains the data and additional information required to perform a encrypt operation. See *JxfsPINCryptoData* specification).

Event

OperationCompleteEvent

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_ENCRYPT
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_KEY_NOT_FOUND The specified key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by this key JXFS_E_PIN_LENGTH_ERROR The length of the start value specified is not supported. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	A **JxfsPINCryptoResult** object. It contains the results of the encryption.

Exceptions

Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PIN_CRYPTNOTSUPPORTED	The encryption method is not supported.

generateMAC Method

Syntax

identificationID generateMAC (JxfsPINMACData macData) throws JxfsException;

Description

Generates a MAC data with the currently selected algorithm.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINMACData	I	macData	Contains the data and additional information required to perform a decrypt operation. See *JxfsPINMACData* specification).

Event

OperationCompleteEvent

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_GENMAC
identificationID	Identification Id of complete operation.
result	

	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_KEY_NOT_FOUND The specified key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by this key JXFS_E_PIN_LENGTH_ERROR The length of the start value specified is not supported. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	A **JxfsPINCryptoResult** object. It contains the generated MAC.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	Meaning
JXFS_E_PIN_CRYPTNOTSUPPORTED	The encryption method is not supported.

getKeyInfo Method

Syntax ***JxfsPINKeyDetail getKeyInfo (java.lang.String keyName) throws JxfsException;***

Description Retrieves information about a given key
Returns a *JxfsPINKeyDetail* object with the requested info.

Parameter

Type	IO	Name	Meaning
String	I	keyName	Name of the key to be queried.

Event No additional events are generated:

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	Meaning
JXFS_E_PIN_KEY_NOT_FOUND	The specified key was not found.

getKeyNameList Method

Syntax ***java.util.Vector getKeyNameList () throws JxfsException;***

Description Retrieves the list of keys names used by the device.
Returns a vector of strings with the name of all keys stored in the device.

Event No additional events are generated.

Exceptions No additional exceptions are generated.

importKey Method

Syntax ***identificationID importKey (JxfsPINKeyToImport keyToImport, boolean lastOrOnlyPart) throws JxfsException;***

Description Loads a key or part of a key into the encryption module. The key can be passed in clear text mode or encrypted with an accompanying "key encryption key".

The imported key is imported into the encryption module and is used for cryptographic operations.

The key may be loaded in parts.

Parameter	Type	IO	Name	Meaning
	JxfsPINKeyToImport	I	keyToImport	Contains the data required to import the key (see *JxfsPINKeyToImport* specification).
	boolean	I	lastOrOnlyPart	If true, key import is finished.

Event

OperationCompleteEvent

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	Value
operationID	JXFS_O_PIN_IMPORTKEY
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_KEY_NOT_FOUND The specified key encryption key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by the specified key. JXFS_E_PIN_DUPLICATE_KEY A key exists with the specified name and cannot be overwritten. JXFS_E_PIN_LENGTH_ERROR The length of the key value specified is not supported. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	A **JxfsPINKeyVerificationData** Object.

Status Event

If the completion of this operation results in an updated key in device's table key, then the J/XFS PIN Keypad device control will fire a StatusEvent to al registered listeners:

Field	Value
status	JXFS_S_PIN_KEY A new key has been loaded/imported into the device's key table.
details	A **JxfsPINKeyDetail** object containing information about the added key.

Exceptions No additional exceptions are generated.

initialize Method

Syntax ***identificationID initialize (byte[ ] id, byte[ ] key) throws JxfsException;***

Description Clears all loaded or imported keys from device's key table.

Usually this operation is invoked by an operator task and not by the application program.

During initialization, an optional encrypted Id key can be stored in the device. The Id key and the corresponding encryption key can be passed as parameters; if not, they are generated automatically by the encryption module. The encrypted Id is returned to the application and serves, if supported (see idKey property), as authorization for the key import function.
This function also resets the HSM terminal data, except session key index and trace number.
This function resets all certificate data and authentication public/private keys (including those replaced by generateRSAKeyPair) back to their initial states at the time of production. Any keys installed during production, which have been permanently replaced, will not be reset. Any Verification certificates that may have been loaded must be reloaded. The Certificate state will remain the same, but the certificate must be re-imported.

Parameter

Type	**IO**	**Name**	**Meaning**
byte[ ]	I	id	ID Key. This byte array is encrypted under *key* and stored into the device. **Null** if not required.
byte[ ]	I	key	Encryption key of *id*. It is also stored into the device. If **null**, *id* is in clear mode.

Event **OperationCompleteEvent**
When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent event listeners.

Field	**Value**
operationID	JXFS_O_PIN_INITIALIZE
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	A **JxfsPINInitialization** Object.

Exceptions No additional exceptions are generated.

importEMVRSAPublicKey Method

Syntax ***identificationID importEMVRSAPublicKey (JxfsPINEMVRSAKeyToImport RSAkeyToImport,) throws JxfsException;***

Description Loads a EMV RSA public key into the encryption module. The RSA key can be provided either by a Certification Authority or by the EMV application in the Chipcard.
This method is similar to the importKey method, but it is specifically designed to address the key formats and security features defined by EMV. Mainly the extensive use of “signed certificate” or “EMV certificate” (which is a compromise between signature and a pure

certificate) to provide the public key is taken in account.
The device services is responsible for all EMV public key import validation. Once loaded, the service provider is not responsible for key/certificate expiry, this is an application responsibility.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINEMVRSAKeyToImport	I	EMVRSAkeyToImport	Contains the data required to import the key (see *JxfsPINEMVRSAKeyToImport* specification).

Event

OperationCompleteEvent
When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_IMPORTEMVRSAKEY
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_KEY_NOT_FOUND The specified key encryption key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by the specified key. JXFS_E_PIN_DUPLICATE_KEY A key exists with the specified name and cannot be overwritten. JXFS_E_PIN_LENGTH_ERROR The length of the key value specified is not supported. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason. JXFS_E_PIN_EMV_VERIFY_FAILED The verification of the key failed and the key is discarded JXFS_E_PIN_KEYRAM_FULL There is no space left in the key RAM for a key of the specified type
data	A **JxfsPINKeyVerificationData** Object.

Status Event
If the completion of this operation results in an updated key in device's table key, then the J/XFS PIN Keypad device control will fire a StatusEvent to all registered listeners:

Field	**Value**
status	JXFS_S_PIN_KEY A new key has been loaded/imported into the device's key table.
details	A **JxfsPINKeyDetail** object containing information about the added key.

Exceptions No additional exceptions are generated.

computeSHA1Digest Method

Syntax	***identificationID computeSHA1Digest (JxfsSHA1Data SHA1Data) throws JxfsException;***
Description	Computes a digest using a SHA-1 algorithm on a stream of data. This method can be used to verify the EMV Static Data Authentication or the Dynamic Data Authentication

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsSHA1Data	I	SHA1Data	Contains the data and the length of data to be hashed (See J*xfsSHA1Data* specification).

Event

OperationCompleteEvent
When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_SHA1_DIGEST
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by this key JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	A **JxfsSHA1Data** object. It contains the result of SHA1 algorithm.

Exceptions

Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PIN_CRYPTNOTSUPPORTED	The encryption method is not supported.

deleteKey Method

Syntax	***identificationID deleteKey (java.lang.String keyName) throws JxfsException;***
Description	deletes a key from the encryption module which was previously stored

Parameter

Type	**IO**	**Name**	**Meaning**
java.lang.String	I	keyName	Contains the name of the key to be deleted from the encryption module.

Event

OperationCompleteEvent
When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_DELETE
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled.

	JXFS_E_PIN_KEY_NOT_FOUND The specified key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The specified use is not supported by this key JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	none

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PIN_CRYPTNOTSUPPORTED	The encryption method is not supported.

importRSAPublicKey Method

Syntax ***identificationID importRSApublicKey (JxfsPINImportRSAPublicKey RSAPublicKeyToImport, boolean lastOrOnlyPart) throws JxfsException;***

Description Loads a RSA public key part into the encryption module and will be used for cryptographic operations. The key can be passed in clear text mode or encrypted with an accompanying "key encryption key".

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINImportRSAPublicKey	I	RSAPubliKeyToImport	Contains the data required to import the key (see JxfsPINImportRSAPublicKey specification).
boolean	I	lastOrOnlyPart	If true, key import is finished.

Event **OperationCompleteEvent**

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_IMPORTRSAPUBLICKEY
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_KEY_NOT_FOUND The specified key encryption key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The key does not support the specified use. JXFS_E_PIN_DUPLICATE_KEY A key exists with the specified name and cannot be overwritten. JXFS_E_PIN_LENGTH_ERROR The length of the key value specified is not supported.

	JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	A ***JxfsPINRSAKeyVerificationData*** Object.

Status Event

If the completion of this operation results in an updated key in device's table key, then the J/XFS PIN Keypad device control will fire a StatusEvent to all registered listeners:

Field	**Value**
status	JXFS_S_PIN_KEY A new key has been loaded/imported into the device's key table.
details	A ***JxfsPINKeyDetail*** object containing information about the added key.

Exceptions No additional exceptions are generated.

exportRSAPublicKey Method

Syntax ***identificationID exportRSAPublicKey (JxfsPINExportRSAPublicKey RSAPublicKeyToExport) throws JxfsException;***

Description This command will export the RSA Public key associated with this PIN device. The RSA public key to export is either the issuer key pair or a previously generated RSA key pair.
Other secure devices will use this key to communicate information securely with this PIN device, using RSA public key encryption.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINExportRSAPublicKey	I	RSAPublicKeyToExport	Contains the data required to export the RSA public key

Event **OperationCompleteEvent**

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_EXPORTRSAPUBLICKEY
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully.
	JXFS_E_CANCELLED Operation was cancelled.
	JXFS_E_PIN_KEY_NOT_FOUND The specified key encryption key was not found.
	JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded.
	JXFS_E_PIN_USE_VIOLATION The key does not support the specified use.
	JXFS_E_PIN_DUPLICATE_KEY A key exists with the specified name and cannot be overwritten.

		JXFS_E_PIN_LENGTH_ERROR The length of the key value specified is not supported. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason. JXFS_E_PIN_NO_RSA_KEY_PAIR The encryption module does not have a RSA private key.
	data	A ***JxfsPINExportedRSAPublicKey*** Object.

Status Event

If the completion of this operation results in an updated key in device's table key, then the J/XFS PIN Keypad device control will fire a StatusEvent to all registered listeners:

Field	**Value**
status	JXFS_S_PIN_KEY A new key has been loaded/imported into the device's key table.
details	A ***JxfsPINKeyDetail*** object containing information about the added key.

Exceptions No additional exceptions are generated.

importRSADESEncipheredPublicKey Method

Syntax ***identificationID importRSADESEncipheredPublicKey (JxfsPINImportRSADESEncipheredPublicKey RSADESEncipheredPublicKeyToImport) throws JxfsException;***

Description This command is used to load a Symmetric Key that is either a single or double DES length key into the encryptor. The key passed by the application is loaded in the encryption module, the (optional) signature & hash are used during validation, the key is extracted using the device's RSA Private Key, and is then stored. The loaded key will be discarded at any stage if any of the above fails.

The random number previously obtained from the ***startKeyExchange*** command and sent to the host is included in the signed data. This random number (when present) is verified during the load process. This command ends the Key Exchange process.

If a Signature algorithm is specified that is not supported by the PIN DS, then the message will not be decrypted and the command fails

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINImportRSADESEncipheredPublicKey	I	RSADESEncipheredPublicKeyToImport	Contains the data of the enciphered imported key.

Event **OperationCompleteEvent**

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_IMPORTRSADESENCIPHEREDPUBLICKEY
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled.

		JXFS_E_PIN_KEY_NOT_FOUND The specified key encryption key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The key does not support the specified use. JXFS_E_PIN_DUPLICATE_KEY A key exists with the specified name and cannot be overwritten. JXFS_E_PIN_LENGTH_ERROR The length of the key value specified is not supported. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason. JXFS_E_PIN_NO_KEY_RAM There are no space left in the key RAM for a key or the specified key JXFS_E_PIN_NO_HASH SUPPORT The DS does not support Hash computation. JXFS_E_PIN_ERR_HASH The imported key failed its hash verification. It is not stored in the PIN JXFS_E_PIN_NO_SIGNATURE_SUP PORT The DS does not support Signature computation. The key was discarded JXFS_E_PIN_ERR_SIGNATURE The imported key failed its signature verification. It is not stored in the PIN
	data	A ***JxfsPINRSADESkeyVerificationData*** Object.

Status Event
If the completion of this operation results in an updated key in device's table key, then the J/XFS PIN Keypad device control will fire a StatusEvent to all registered listeners:

Field	**Value**
status	JXFS_S_PIN_KEY A new key has been loaded/imported into the device's key table.
details	A ***JxfsPINKeyDetail*** object containing information about the added key.

Exceptions No additional exceptions are generated.

exportRSADESEncipheredPublicKey Method

Syntax ***identificationID*** **exportRSADESEncipheredPublicKey *()*** ***throws JxfsException;***

Description This command will export the RSA DES enciphered Public key associated with this PIN device. Other secure devices will use this key to communicate information securely with this PIN device, using RSA public\private key encryption.

Parameter	**Type**	**IO**	**Name**	**Meaning**

Event **OperationCompleteEvent**

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_EXPORTRSADESENCIPHERDPUBLICKEY
identificationID	Identification Id of complete operation.
result	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_KEY_NOT_FOUND The specified key encryption key was not found. JXFS_E_PIN_KEY_NO_VALUE The specified key is not loaded. JXFS_E_PIN_USE_VIOLATION The key does not support the specified use. JXFS_E_PIN_DUPLICATE_KEY A key exists with the specified name and cannot be overwritten. JXFS_E_PIN_LENGTH_ERROR The length of the key value specified is not supported. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	A ***JxfsPINExportRSADESEncipheredPublicKey*** Object.

Exceptions No additional exceptions are generated.

generateRSAKeyPair Method

Syntax ***identificationID* generateRSAKeyPair *(JxfsPINGenerateRSAKeyPair RSAKeyPairToGenerate) throws JxfsException;***

Description This command will generate a new RSA key pair.

Parameter

Type	**IO**	**Name**	**Meaning**
JxfsPINGenerateRSAKeyPair	I	RSAKeyPairToGenerate	Contains the data of the generated RSA Key pair.

Event **OperationCompleteEvent**

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_GENERATERSAKEYPAIR
identificationID	Identification Id of complete operation.
result	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled.

		JXFS_E_PIN_USE_VIOLATION The key does not support the specified use. JXFS_E_PIN_LENGTH_ERROR The length of the key value specified is not supported. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
	data	none.

Status Event

If the completion of this operation results in an updated key in device's table key, then the J/XFS PIN Keypad device control will fire a StatusEvent to all registered listeners:

Field	**Value**
status	JXFS_S_PIN_KEY A new key has been loaded/imported into the device's key table.
details	A ***JxfsPINKeyDetail*** object containing information about the added key.

Exceptions No additional exceptions are generated.

exportPINId Method

Syntax ***identificationID exportPINId () throws JxfsException;***

Description This command is used to retrieve the Security Item that uniquely identifies the PIN device. This value may be used to uniquely identify a PIN device and therefore confer trust upon any key or data obtained from this device.

Event **OperationCompleteEvent**

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_EXPORTPINID
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	***JxfsPINExportId***

Status Event

If the completion of this operation results in an updated key in device's table key, then the J/XFS PIN Keypad device control will fire a StatusEvent to all registered listeners:

Field	**Value**
status	JXFS_S_PIN_KEY A new key has been loaded/imported into the device's key table.
details	A ***JxfsPINKeyDetail*** object containing information about the added key.

Exceptions No additional exceptions are generated.

importCertificate Method

Syntax ***identificationID*** **importCertificate** ***(byte []CertificateToImport) throws JxfsException;***

Description This command is used to load a certificate provided by a Certification Authority (CA) to be used for remote key loading. This command can be called only once, if there are no plans for a new CA to take over the duties. If a new CA does take over the duties, then this command should be called after the ***replaceCertificate*** method. The type of certificate (Primary or Secondary) to be loaded will be embedded within the actual certificate structure.

Parameter

Type	IO	Name	Meaning
byte [ ]	I	CertificateToImport	Contains the certificate that is to be loaded. This data should be in a binary encoded PKCS #7 format containing certificate data represented in DER encoded ASN.1 notation

Event **OperationCompleteEvent**

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	Value
operationID	JXFS_O_PIN_IMPORTCERTIFICATE
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_USE_VIOLATION The key does not support the specified use. JXFS_E_PIN_INVALID_FORMAT The format of the message is invalid JXFS_E_PIN_INVALID_STATE The certificate module is in a state in which the request is invalid. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	***byte []*** SHA-1 Thumb print value. This data should be in a binary encoded PKCS #7 format containing certificate data represented in DER encoded ASN.1 notation.

Status Event

If the completion of this operation results in an updated key in device's table key, then the J/XFS PIN Keypad device control will fire a StatusEvent to all registered listeners:

Field	Value
status	JXFS_S_PIN_KEY A new key has been loaded/imported into the device's key table.
details	A ***JxfsPINKeyDetail*** object containing information about the added key.

Exceptions No additional exceptions are generated.

exportCertificate Method

Syntax ***identificationID* exportCertificate *(JxfsPINCertificateKeyType certificateKeyType) throws JxfsException;***

Description This command is used to read out from the encryptor the certificate that was signed by a certification Authority (CA). This certificate is sent to the host.

Parameter

Type	IO	Name	Meaning
JxfsPINCertificateKeyType	I	certificateKeyType	Specifies which public key to be used.

Event **OperationCompleteEvent**

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	Value
operationID	JXFS_O_PIN_EXPORTCERTIFICATE
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_USE_VIOLATION The key does not support the specified use. JXFS_E_PIN_INVALID_FORMAT The format of the message is invalid JXFS_E_PIN_INVALID_STATE The certificate module is in a state in which the request is invalid. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	***JxfsPINExportCertificate*** *object*

Status Event

If the completion of this operation results in an updated key in device's table key, then the J/XFS PIN Keypad device control will fire a StatusEvent to all registered listeners:

Field	Value
status	JXFS_S_PIN_KEY A new key has been loaded/imported into the device's key table.
details	A **JxfsPINKeyDetail** object containing information about the added key.

Exceptions No additional exceptions are generated.

replaceCertificate Method

Syntax ***identificationID* replaceCertificate *(byte [] newCertificate) throws JxfsException;***

Description This operation will replace either the primary or secondary Certificate Authority certificate previously loaded inside the encryptor. After this command is complete, the application should send the ***importCertificate*** and ***exportCertificate*** commands to ensure that the new HOST and the encryptor have both all the information required to perform the remote key loading process.

Parameter

Type	IO	Name	Meaning
byte []	I	newCertificate	Contains the new certificate that is to be loaded. This data should be in a binary encoded

PKCS #7 format containing certificate data represented in DER encoded ASN.1 notation.

Event **OperationCompleteEvent**

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_REPLACECERTIFICATE
identificationID	Identification Id of complete operation.
result	
	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_USE_VIOLATION The key does not support the specified use. JXFS_E_PIN_INVALID_FORMAT The format of the message is invalid JXFS_E_PIN_INVALID_STATE The certificate module is in a state in which the request is invalid. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
data	***byte []*** SHA-1 Thumb print value. This data should be in a binary encoded PKCS #7 format containing certificate data represented in DER encoded ASN.1 notation.

Status Event

If the completion of this operation results in an updated key in device's table key, then the J/XFS PIN Keypad device control will fire a StatusEvent to all registered listeners:

Field	**Value**
status	JXFS_S_PIN_KEY A new key has been loaded/imported into the device's key table.
details	A ***JxfsPINKeyDetail*** object containing information about the added key.

Exceptions No additional exceptions are generated.

startKeyExchange Method

Syntax ***identificationID*** **startKeyExchange** ***() throws JxfsException***

Description This command is used to start the transfer of the host's Key Transport Key. The encryptor generates a random number that will be used to verify tke key Transport message sent by the host.
The key exchange process is ended with the command importRSADESEncipheredPublicKey command.

Event **OperationCompleteEvent**

When the operation completes a *JxfsOperationCompleteEvent* event will be sent by J/XFS PINKeypad Device Control to all registered JxfsOperationCompleteEvent listeners.

Field	**Value**
operationID	JXFS_O_PIN_STARTKEYEXCHANGE
identificationID	Identification Id of complete operation.

	result	
		JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_PIN_ACCESS_DENIED The encryption module is either not initialized or not ready for any vendor specific reason.
	data	***byte []:*** Specifies an 8 bytes randomly generated number created by the encryptor.

Status Event

If the completion of this operation results in an updated key in device's table key, then the J/XFS PIN Keypad device control will fire a StatusEvent to all registered listeners:

Field	**Value**
status	JXFS_S_PIN_KEY A new key has been loaded/imported into the device's key table.
details	A ***JxfsPINKeyDetail*** object containing information about the added key.

Exceptions No additional exceptions are generated.

5 Support Classes

5.1 JxfsPINFKeySet

This class provides properties and methods to query which function keys are supported or are active.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
fk0	boolean	R	
fk1	boolean	R	
fk2	boolean	R	
fk3	boolean	R	
fk4	boolean	R	
fk5	boolean	R	
fk6	boolean	R	
fk7	boolean	R	
fk8	boolean	R	
fk9	boolean	R	
fkEnter	boolean	R	
fkCancel	boolean	R	
fkClear	boolean	R	
fkBackspace	boolean	R	
fkHelp	boolean	R	
fkDecPoint	boolean	R	
fk00	boolean	R	
fk000	boolean	R	

Method	Return	May use after
is*Property*	*Property*	
allFKeys	boolean	
noFKeys	boolean	
JxfsPINFKeySet	(constructor of the class)	

5.1.1 Properties

fk0 .. fk000 Properties (R)

Type ***boolean***
Initial Value FALSE
Description Indicates if related function key is selected.
Note: fk00 and fk000 (hundred's and thousand's keys) are treated as sequences of two and three fk0, respectively.

Value	Meaning
FALSE	Function key is not selected.
TRUE	Function key is selected.

5.1.2 Methods

allFKeys Method

Syntax ***boolean allFKeys ()***
Description Returns TRUE if all properties are set to TRUE.

noFKeys Method

Syntax	***boolean noFKeys ()***
Description	Returns TRUE if all properties are set to FALSE.

JxfsPINFKeySet Constructor

Syntax	***JxfsPINFKeySet (boolean fk0, boolean fk1, ... , boolean fk000)***
Description	Constructor of the class.

5.2 JxfsPINFKeysSelection

This class provides properties and methods to query and select which function keys are active.

Summary

Implements : **Extends :** **JxfsPINFKeySet**

Property	Type	Access	Initialized after
No additional properties.			

Method	Return	May use after
set*Property*	void	
setAllFKeys	void	
setNoFKeys	void	
JxfsPINFKeysSelection	(constructor of the class)	

5.2.1 Properties

No additional properties to those inherited from base class *JxfsPINFKeySet*.

5.2.2 Methods

setAllFKeys Method

Syntax	***void setAllFKeys ()***
Description	Sets all properties to TRUE.

setNoFKeys Method

Syntax	***void setNoFKeys ()***
Description	Sets all properties to FALSE.

JxfsPINFKeysSelection Constructor

Syntax	***JxfsPINFKeysSelection (boolean fk0, ... , boolean fk000)***
Description	Constructor of the class.

5.3 JxfsPINFDKeysSelection

This class provides properties and methods to query and select which function descriptor keys (FDKeys) are active.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
fdk01	boolean	R/W	
fdk02	boolean	R/W	
fdk03	boolean	R/W	
fdk04	boolean	R/W	
fdk05	boolean	R/W	
fdk06	boolean	R/W	
fdk07	boolean	R/W	
fdk08	boolean	R/W	
fdk09	boolean	R/W	
fdk10	boolean	R/W	
fdk11	boolean	R/W	
fdk12	boolean	R/W	
fdk13	boolean	R/W	
fdk14	boolean	R/W	
fdk15	boolean	R/W	
fdk16	boolean	R/W	
fdk17	boolean	R/W	
fdk18	boolean	R/W	
fdk19	boolean	R/W	
fdk20	boolean	R/W	
fdk21	boolean	R/W	
fdk22	boolean	R/W	
fdk23	boolean	R/W	
fdk24	boolean	R/W	
fdk25	boolean	R/W	
fdk26	boolean	R/W	
fdk27	boolean	R/W	
fdk28	boolean	R/W	
fdk29	boolean	R/W	
fdk30	boolean	R/W	
fdk31	boolean	R/W	
fdk32	boolean	R/W	

Method	Return	May use after
is*Property*	*Property*	
set*Property*	void	
allFDKeys	boolean	
noFDKeys	boolean	
setAllFDKeys	void	
setNoFDKeys	void	
JxfsPINFDKeysSelection	(constructor of the class)	

5.3.1 Properties

fdk01 .. fdk32 Properties (R/W)

Type	***boolean***
Initial Value	FALSE
Description	Indicates if related function descriptor key is selected.

Value	Meaning
FALSE	Function descriptor key is not selected.
TRUE	Function descriptor key is selected.

5.3.2 Methods

allFDKeys Method

Syntax	***boolean allFDKeys ()***
Description	Returns TRUE if all properties are set to TRUE.

noFDKeys Method

Syntax	***boolean noFDKeys ()***
Description	Returns TRUE if all properties are set to FALSE.

setAllFDKeys Method

Syntax	***void setAllFDKeys ()***
Description	Sets all properties to TRUE.

setNoFDKeys Method

Syntax	***void setNoFDKeys ()***
Description	Sets all properties to FALSE.

JxfsPINFDKeysSelection Constructor

Syntax	***JxfsPINFDKeysSelection (boolean fdk01, ... , boolean fdk32)***
Description	Constructor of the class.

5.4 JxfsPINFDKey

Summary

The JxfsPINFDKey class contains information about a function descriptor key (FDKey).

Implements : **Extends :** **JxfsType**

Property	**Type**	**Access**	**Initialized after**
keyCode	int	R	
relativeX	int	R	
relativeY	int	R	

Method	**Return**	**May use after**
get*Property*	*Property*	
JxfsPINFDKey	(constructor of the class)	

5.4.1 Properties

keyCode Property (R)

Type	***int***
Description	Specifies the code used for the function descriptor key FDKey. Its value is one of the following:

Value
JXFS_PIN_FK_FDK01
JXFS_PIN_FK_FDK02
JXFS_PIN_FK_FDK03
JXFS_PIN_FK_FDK04
JXFS_PIN_FK_FDK05
JXFS_PIN_FK_FDK06
JXFS_PIN_FK_FDK07
JXFS_PIN_FK_FDK08
JXFS_PIN_FK_FDK09
JXFS_PIN_FK_FDK10
JXFS_PIN_FK_FDK11
JXFS_PIN_FK_FDK12
JXFS_PIN_FK_FDK13
JXFS_PIN_FK_FDK14
JXFS_PIN_FK_FDK15
JXFS_PIN_FK_FDK16
JXFS_PIN_FK_FDK17
JXFS_PIN_FK_FDK18
JXFS_PIN_FK_FDK19
JXFS_PIN_FK_FDK20
JXFS_PIN_FK_FDK21
JXFS_PIN_FK_FDK22
JXFS_PIN_FK_FDK23
JXFS_PIN_FK_FDK24
JXFS_PIN_FK_FDK25
JXFS_PIN_FK_FDK26
JXFS_PIN_FK_FDK27
JXFS_PIN_FK_FDK28
JXFS_PIN_FK_FDK29
JXFS_PIN_FK_FDK30
JXFS_PIN_FK_FDK31
JXFS_PIN_FK_FDK32

relativeX Property (R)

Type	***int***
Description	Specifies the FDKey position relative to the left hand side of the screen expressed as a percentage of the width of the screen. For this, the FDKey position is defined by the point which results from perpendicular projection of the key center onto the edge of the screen.

relativeY Property (R)

Type	***Int***
Description	Specifies the FDKey position relative to the top of the screen expressed as a percentage of the height of the screen. For this, the FDKey position is defined by the point which results from perpendicular projection of the key center onto the edge of the screen.

5.4.2 Methods

JxfsPINFDKey Constructor

Syntax	***JxfsPINFDKey (int keyCode, int relativeX, int relativeY)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Some parameter is out of range.

5.5 JxfsPINReadMode

This class specifies the conditions for PIN keypad data entry when using *readData()* and *secureReadPIN()* methods.

Summary

Implements : **Extends :** *JxfsType*

Property	**Type**	**Access**	**Initialized after**
activeFDKeys	**JxfsPINFDKeysSelection**	R/W	
activeFKeys	**JxfsPINFKeysSelection**	R/W	
terminateFDKeys	**JxfsPINFDKeysSelection**	R/W	
terminateFKeys	**JxfsPINFKeysSelection**	R/W	
autoEnd	boolean	R/W	
beepOnPress	boolean	R/W	
inputMode	int	R/W	
maxLength	int	R/W	
minLength	int	R/W	

Method	**Return**	**May use after**
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINReadMode	(constructor of the class)	

5.5.1 Properties

activeFDKeys Property (R/W)

Type	***JxfsPINFDKeysSelection***
Initial Value	Null until open.
Description	Indicates the set of function descriptor keys (FDKeys) enabled for subsequent input operations.

activeFKeys Property (R/W)

Type	***JxfsPINFKeysSelection***
Initial Value	Null until open.
Description	Indicates the set of function keys enabled for subsequent input operations.

terminateFDKeys Property (R/W)

Type	***JxfsPINFDKeysSelection***
Initial Value	Null until open.
Description	Specifies the set of function descriptor keys (FDKeys) that, if pressed during an input operation, will terminate a data entry. It must be a subset of the set defined by *activeFDKeys.*

terminateFKeys Property (R/W)

Type	***JxfsPINFKeysSelection***
Initial Value	Null until open.
Description	Specifies the set of function keys that, if pressed during an input operation, will terminate a data entry.

It must be a subset of the set defined by *activeFKeys.*

autoEnd Property (R/W)

Type	***boolean***
Initial Value	FALSE
Description	Indicates the criteria used to terminate subsequent input operations.

If *maxLength* is set to 0, this property is ignored and input is only terminated by a termination key (see *terminateFKeys* and *terminateFDKeys* properties).

Value	**Meaning**
TRUE	PIN entry terminates when the maximun number of digits are entered (*maxLength* property).
FALSE	PIN entry terminates when a termination key (*terminateFKey*s and *terminateFDKeys* properties) has been pressed. In this case, when *maxLength* is reached, numeric keys are disabled by the device service.

beepOnPress Property (R/W)

Type	***boolean***
Initial Value	FALSE
Description	Specifies if the device must generate an audible sound at every key press or not.

Value	**Meaning**
FALSE	The device must not beep.
TRUE	The device must beep.

inputMode Property (R/W)

Type	***int***
Initial Value	JXFS_PIN_INPUT_COOKED
Description	Specifies the input mode to be used in subsequent input operations.

Value	**Meaning**
JXFS_PIN_INPUT_RAW	Each key pressed during an input operation will generate an intermediate event. These events will contain information about pressed keys.
JXFS_PIN_INPUT_COOKED	No intermediate events per key pressed are generated. Data entered during an input operation is provided in the *OperationCompleteEvent* event.

maxLength Property (R/W)

Type	***int***
Initial Value	8
Description	Specifies the maximum number of digits which can be entered in an input operation.

If autoEnd is set to TRUE, the input operation ends when this maximun number of digits has been entered.

If it is set to zero, the input operation does not end until a termination key is pressed (see *terminateKeys* and *terminateFDKeys* properties). If no termination keys are specified, the input operation will not terminate until a *cancel()* operation is issued.

minLength Property (R/W)

Type	***int***
Initial Value	1
Description	Specifies the minimum number of digits which must be entered for a valid input operation. A value of JXFS_PIN_NO_MINUMUM_LENGTH (zero) indicates no minimum PIN length verification.

5.5.2 Methods

JxfsPINReadMode Constructor

Syntax ***JxfsPINReadMode (JxfsPINFDKeysSelection activeFDKeys, JxfsPINFKeysSelection activeFKeys, JxfsPINFDKeysSelection terminateFDKeys, JxfsPINFKeysSelection terminateFKeys, boolean autoEnd, boolean beepOnPress, int inputMode, int maxLength, int minLength)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	Meaning
JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *activeFDKeys* is null. *activeFKeys* is null. *terminateFDKeys* is null. *terminateFKeys* is null. *inputMode* is not one of the listed values. *maxLength* is less than *minLength.* *minLength* is negative.

5.6 JxfsPINReadMode2

This class specifies extended conditions for PIN keypad data entry when using the *readData()* and *secureReadPIN()* methods.

Summary

Implements : **Extends :** *JxfsPINReadMode*

Property	Type	Access	Initialized after
eventOnStart	boolean	R/W	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINReadMode2	(constructor of the class)	

5.6.1 Properties

eventOnStart Property (R/W)

Type	***boolean***
Initial Value	FALSE.
Description	Specifies if the service must send an intermediate event when the device is ready to accept user entered data.

Value	**Meaning**
FALSE	The service must not send the event
TRUE	The service must send the event

5.6.2 Methods

JxfsPINReadMode2 Constructor

Syntax	***JxfsPINReadMode2 (JxfsPINFDKeysSelection activeFDKeys, JxfsPINFKeysSelection activeFKeys, JxfsPINFDKeysSelection terminateFDKeys, JxfsPINFKeysSelection terminateFKeys, boolean autoEnd, boolean beepOnPress, int inputMode, int maxLength, int minLength, boolean eventOnStart)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *activeFDKeys* is null. *activeFKeys* is null. *terminateFDKeys* is null. *terminateFKeys* is null. *inputMode* is not one of the listed values. *maxLength* is less than *minLength.* *minLength* is negative.

5.7 JxfsPINPressedKey

This class contains the data associated to a pressed key during an input operation.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
keyCode	int	R	
keyType	int	R	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINPressedKey	(constructor of the class)	

5.7.1 Properties

keyCode Property (R)

Type ***Int***

Description Code of key.

Value	Meaning
JXFS_PIN_FK_NONE	If result of a *secureReadPIN()* operation and the key is a numeric function key. This value may be used to output substitution signs on a display.
JXFS_PIN_FK_FDK01	
JXFS_PIN_FK_FDK02	
JXFS_PIN_FK_FDK03	
JXFS_PIN_FK_FDK04	
JXFS_PIN_FK_FDK05	
JXFS_PIN_FK_FDK06	
JXFS_PIN_FK_FDK07	
JXFS_PIN_FK_FDK08	
JXFS_PIN_FK_FDK09	
JXFS_PIN_FK_FDK10	
JXFS_PIN_FK_FDK11	
JXFS_PIN_FK_FDK12	
JXFS_PIN_FK_FDK13	
JXFS_PIN_FK_FDK14	
JXFS_PIN_FK_FDK15	
JXFS_PIN_FK_FDK16	
JXFS_PIN_FK_FDK17	
JXFS_PIN_FK_FDK18	
JXFS_PIN_FK_FDK19	
JXFS_PIN_FK_FDK20	
JXFS_PIN_FK_FDK21	
JXFS_PIN_FK_FDK22	
JXFS_PIN_FK_FDK23	
JXFS_PIN_FK_FDK24	
JXFS_PIN_FK_FDK25	
JXFS_PIN_FK_FDK26	
JXFS_PIN_FK_FDK27	
JXFS_PIN_FK_FDK28	
JXFS_PIN_FK_FDK29	
JXFS_PIN_FK_FDK30	
JXFS_PIN_FK_FDK31	
JXFS_PIN_FK_FDK32	

JXFS_PIN_FK_0
JXFS_PIN_FK_1
JXFS_PIN_FK_2
JXFS_PIN_FK_3
JXFS_PIN_FK_4
JXFS_PIN_FK_5
JXFS_PIN_FK_6
JXFS_PIN_FK_7
JXFS_PIN_FK_8
JXFS_PIN_FK_9
JXFS_PIN_FK_ENTER
JXFS_PIN_FK_CANCEL
JXFS_PIN_FK_CLEAR
JXFS_PIN_FK_BACKSPACE
JXFS_PIN_FK_HELP
JXFS_PIN_FK_DECPOINT
JXFS_PIN_FK_00
JXFS_PIN_FK_000

keyType Property (R)

Type ***int***

Description Type of key pressed

It can be one of the following values:

Value	**Meaning**
JXFS_PIN_KP_FUNCTION	Function key.
JXFS_PIN_KP_FDKEY	Function descriptor key (FDKey).

5.7.2 Methods

JxfsPINPressedKey Constructor

Syntax ***JxfsPINPressedKey (int keyCode, int keyType)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *keyCode* is not one of the listed values. *keyType* is not one of the listed values.

5.8 JxfsPINReadData

This class contains the data returned by an *OperationCompleteEvent* event for *readData*() and *secureReadPIN()* operations.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
endReason	int	R	
pinLength	int	R	
pressedKeys	java.util.Vector	R	
readData	java.lang.String	R	
terminationKey	int	R	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINReadData	(constructor of the class)	

5.8.1 Properties

endReason Property (R)

Type *int*

Description Indicates the input operation termination reason.

It can be one of the following values:

Value	Meaning
JXFS_PIN_COMP_AUTO	Input operation terminated because *maxLength* was reached.
JXFS_PIN_COMP_FK	A termination key was pressed.
JXFS_PIN_COMP_FDKEY	A termination FDKey was pressed

pinLength Property (R)

Type *int*

Description If *inputMode* property is set to JXFS_PIN_INPUT_RAW, it contains the count of keys pressed.
If *inputMode* property is set to JXFS_PIN_INPUT_COOKED, it contains the count of digits entered.

pressedKeys Property (R)

Type *java.util.Vector*

Description Vector of **JxfsPINPressedKey** objects. It represents the list of all the keys pressed during the input operation.

If *inputMode* Property was set to JXFS_PIN_INPUT_RAW this property is optional and can be set to null.

readData Property (R)

Type *java.lang.String*

Description Cooked data entered in input operation.

Value	Meaning

null	• if result of a *secureReadPIN()* operation. • if result of a *readData()* operation and *inputMode* Property was set to JXFS_PIN_INPUT_RAW.
Non formatted string representation of numeric value entered. Function keys are omitted.	• if result of a *readData()* operation and *inputMode* Property was set to JXFS_PIN_INPUT_COOKED.

terminationKey Property (R)

Type ***int***

Description Code of termination function key or FDKey if end reason was JXFS_PIN_COMP_FK or JXFS_PIN_COMP_FDKEY.

If termination reason was JXFS_PIN_COMP_AUTO, it is set to JXFS_PIN_FK_NONE.

5.8.2 Methods

JxfsPINReadData Constructor

Syntax ***JxfsPINReadData (int endReason, int pinLength, java.util.Vector pressedKeys, java.lang.String readData, int terminationKey)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *endReason* is not one of the listed values. *pinLength* is negative. *pressedKeys* is null and inputMode is JXFS_PIN_INPUT_COOKED. *readData* is null and inputMode is JXFS_PIN_INPUT_COOKED. *terminationKey* has an invalid value.

5.9 JxfsPINFormats

This class provides properties and methods to query which PIN formats are supported by a PIN device service.

Summary

Implements :	**Extends :**	**JxfsType**

Property	Type	Access	Initialized after
fmt3624	boolean	R	
fmtANSI	boolean	R	
fmtISO0	boolean	R	
fmtISO1	boolean	R	
fmtEC12	boolean	R	
fmtEC13	boolean	R	
fmtEC13_Rand	boolean	R	
fmtVISA	boolean	R	
fmtDiebold	boolean	R	
fmtDieboldC0	boolean	R	
fmtEMV	boolean	R	

Method	Return	May use after
is*Property*	*Property*	
JxfsPINFormats	(constructor of the class)	

5.9.1 Properties

fmt3624 Property (R)

Type	***boolean***	
Initial Value	Depends on device	
Description	Indicates if the device supports the format: PIN left justified, filled with padding characters, PIN length 4-16 digits.	
	Value	**Meaning**
	FALSE	Format is not supported.
	TRUE	Format is supported.

fmtANSI Property (R)

Type	***boolean***	
Initial Value	Depends on device	
Description	Indicates if the device supports the format: PIN is preceded by 0x00 and the length of the PIN (0x04 to 0x0C), filled with padding character 0x0F to the right, PIN length 4-12 digits, XORed with PAN (Primary Account Number, minimum 12 digits without check number).	
	Value	**Meaning**
	FALSE	Format is not supported.
	TRUE	Format is supported.

fmtISO0 Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports the format: PIN is preceded by 0x00 and the length of the PIN (0x04 to 0x0C), filled with padding character 0x0F to the right, PIN length 4-12 digits, XORed with PAN (Primary

Account Number, no minimum length specified, missing digits are filled with 0x00).

Value	**Meaning**
FALSE	Format is not supported.
TRUE	Format is supported.

fmtISO1 Property (R)

Type *boolean*

Initial Value Depends on device

Description Indicates if the device supports the format: PIN is preceded by 0x01 and the length of the PIN (0x04 to 0x0C), padding characters are taken from a transaction field (10 digits).

Value	**Meaning**
FALSE	Format is not supported.
TRUE	Format is supported.

fmtEC12 Property (R)

Type *boolean*

Initial Value Depends on device

Description Indicates if the device supports the format: similar to fmt3624, PIN only 4 digits.

Value	**Meaning**
FALSE	Format is not supported.
TRUE	Format is supported.

fmtEC13 Property (R)

Type *boolean*

Initial Value Depends on device

Description Indicates if the device supports the format: PIN is preceded by the length (digit), PIN length 4-6 digits, padded with 0x00.

Value	**Meaning**
FALSE	Format is not supported.
TRUE	Format is supported.

fmtEC13_Rand Property (R)

Type *boolean*

Initial Value Depends on device

Description Indicates if the device supports the format: PIN is preceded by the length (digit), PIN length 4-6 digits, padded with random data.

Value	**Meaning**
FALSE	Format is not supported.
TRUE	Format is supported.

fmtVISA Property (R)

Type *boolean*

Initial Value Depends on device

Description Indicates if the device supports the format: same as fmtEC13.

Value	**Meaning**
FALSE	Format is not supported.

TRUE	Format is supported.

fmtDiebold (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports the format: PIN is padded with the padding character and may be not encrypted, single encrypted or double encrypted.

Value	Meaning
FALSE	Format is not supported.
TRUE	Format is supported.

fmtDieboldC0 (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports the format: PIN is preceded by the two-digit coordination number, padded with the padding character and may be not encrypted, single encrypted or double encrypted.

Value	Meaning
FALSE	Format is not supported.
TRUE	Format is supported.

fmtEMV

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports the EMV PIN format: PIN is preceded by 0x02 and the length of the PIN (0x04 to 0x0C), padded with the padding character 0x0F to the right. PIN is formatted up to 248 bytes according to EMV specification V 4.0and finally. PIN is encrypted with a RSA key.

Value	Meaning
FALSE	Format is not supported.
TRUE	Format is supported.

5.9.2 Methods

JxfsPINFormats Constructor

Syntax	***JxfsPINFormats (boolean fmt3624, boolean fmtANSI, boolean fmtSO0, boolean fmtSO1, boolean fmtEC12, boolean fmtEC13, boolean fmtEC13_Rand, boolean fmtVISA, boolean fmtDiebold, boolean fmtDieboldC0)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	Meaning
JXFS_E_PARAMETER_INVALID	All the parameters are false.

JxfsPINFormats Constructor

Syntax	***JxfsPINFormats (boolean fmt3624, boolean fmtANSI, boolean fmtSO0, boolean fmtSO1, boolean fmtEC12, boolean fmtEC13, boolean fmtEC13_Rand, boolean fmtVISA, boolean fmtDiebold, boolean fmtDieboldC0, boolean fmtEMV)***

Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	All the parameters are false.

5.10 JxfsPINValidationAlgorithms

This class provides properties and methods to query which algorithms for PIN validation are supported by a PIN device service.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
valDES	boolean	R	
valEC	boolean	R	
valVISA	boolean	R	
valDESOffset	boolean	R	
valEMVRSA	boolean	R	

Method	Return	May use after
is*Property*	*Property*	
JxfsPINValidationAlgorithms	(constructor of the class)	

5.10.1 Properties

valDES Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports DES algorithm for PIN validation.

Value	**Meaning**
FALSE	Algorithm is not supported.
TRUE	Algorithm is supported.

valEC Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports EUROCHEQUE algorithm for PIN validation.

Value	**Meaning**
FALSE	Algorithm is not supported.
TRUE	Algorithm is supported.

valVISA Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports VISA algorithm for PIN validation.

Value	**Meaning**
FALSE	Algorithm is not supported.
TRUE	Algorithm is supported.

valDESOffset Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports DES offset generation algorithm.

Value	Meaning
FALSE	Offset generation is not supported.
TRUE	Offset generation is supported.

valEMVRSA Property (R)

Type	*boolean*
Initial Value	Depends on device
Description	Indicates if the device supports EMV RSA algorithm for PIN generation

Value	Meaning
FALSE	EMV RSA algorithm is not supported.
TRUE	EMV RSA algorithm is supported.

5.10.2 Methods

JxfsPINValidationAlgorithms Constructor

Syntax	***JxfsPINValidationAlgorithms (boolean valDES, boolean valEC, boolean valVISA, boolean valDESOffset)***
Description	Constructor of the class.

JxfsPINValidationAlgorithms Constructor

Syntax	***JxfsPINValidationAlgorithms (boolean valDES, boolean valEC, boolean valVISA, boolean valDESOffset, boolean valEMVRSA)***
Description	Constructor of the class.

5.11 JxfsPINChipPresentationModes

This class provides properties and methods to query which presentation algorithms for PIN chip validation are supported by a PIN device service.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
presentClear	boolean	R	
presentEMVRSAEnciphered	boolean	R	

Method	Return	May use after
is*Property*	*Property*	
JxfsPINChipPresentationModes	(constructor of the class)	

5.11.1 Properties

presentClear Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports presentation of a clear text PIN to a chip card.

Value	**Meaning**
FALSE	Presentation algorithm is not supported.
TRUE	Presentation algorithm is supported.

PresentEMVRSAEnciphered Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports presentation of a EMV RSA enciphered PIN Block to the chip card

Value	**Meaning**
FALSE	EMV RSA enciphered presentation algoritm is not supported.
TRUE	EMV RSA enciphered presentation algoritm is supported.

5.11.2 Methods

JxfsPINChipPresentationModes Constructor

Syntax	***JxfsPINChipPresentationModes (boolean presentClear)***
Description	Constructor of the class.

JxfsPINChipPresentationModes Constructor

Syntax	***JxfsPINChipPresentationModes (boolean presentClear, boolean PresentEMVRSAEnciphered)***

5.12 JxfsPINValidationData

Abstract class.

The J/XFS PIN Validation Data is the root of a hierarchy of data objects that contain data for PIN verification and used in *validatePIN(), createOffset(), createPINBlock(), validatePINSecure(), createOffsetSecure(), createPINBlockSecure()* methods of JxfsSecurePINKeypad Device Control class.

Summary

Implements : **Extends : JxfsType**

Property	Type	Access	Initialized after
validationAlgorithm	int	R	
keyName	java.lang.String	R/W	
keyEncrKey	byte[ ]	R/W	
validationTrackNumber	int	R/W	
validationLength	int	R/W	
validationIndex	int	R/W	
offsetTrackNumber	int	R/W	
offsetLength	int	R/W	
offsetIndex	int	R/W	
ejectCurrent	boolean	R/W	
ejectWhenComplete	boolean	R/W	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	

5.12.1 Properties

validationAlgorithm Property (R)

Type *int*

Description Validation algorithm for which this object is intended to be used. Set by the constructor of each of the specific subclasses of JxfsPINValidationData to one of the following values:

Value	**Meaning**
JXFS_PIN_VAL_DES	DES PIN validation.
JXFS_PIN_VAL_EC	EUROCHEQUE PIN validation.
JXFS_PIN_VAL_VISA	VISA PIN validation.

keyName Property (R/W)

Type *java.lang.String*

Description Name of the key to be used by the algorithms.
If *keyEncrKey* property is other than **null**, then this key is used to decrypt the keyEncrKey encrypted key and its result is used instead.
If *keyEncrKey* property is **null**, then this key is directly used.

For *JxfsPinBlockData* subclass, it specifies the name of the key used to encrypt the formatted PIN for the first time, or **null** if no encryption is required..

keyEncrKey Property (R/W)

Type *byte[ ]*

Description Optional encrypted (under *keyName*) key to be used for PIN validation.

For *JxfsPinBlockData* subclass, it specifies the name of the key used to format the once encrypted formatted PIN, or **null** if no second encryption is required.

validationTrackNumber Property (R/W)

Type	***int***
Description	Track where validation data is located. Optional property.

validationLength Property (R/W)

Type	***int***
Description	Length of validation data. Optional property.

validationIndex Property (R/W)

Type	***int***
Description	Location of validation data from index zero. Optional property.

offsetTrackNumber Property (R/W)

Type	***int***
Description	Track where offset data is located. Optional property.

offsetLength Property (R/W)

Type	***int***
Description	Length of offset data. Optional property.

offsetIndex Property (R/W)

Type	***int***
Description	Location of offset data from index zero. Optional property.

ejectCurrent Property (R/W)

Type	***boolean***
Description	Set true to eject any card currently in reader. Optional property.

ejectWhenComplete Property (R/W)

Type	***boolean***
Description	Set true to eject card on completion. Optional property.

5.12.2 Exceptions

Exception JXFS_E_PARAMETER_INVALID is thrown by the setter methods in the following cases:

- The value for an int property is negative.

5.13 JxfsPINValidationDataForDES

Class that contains data required for DES PIN validation.

Summary

Implements : **Extends :** *JxfsPINValidationData*

Property	Type	Access	Initialized after
decimalTable	byte[ ]	R/W	
maxPIN	int	R/W	
noLeadingZero	boolean	R/W	
offset	byte[ ]	R/W	
offsetUsed	boolean	R/W	
paddingChar	byte	R/W	
validationData	byte [ ]	R/W	
validationDigits	int	R/W	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINValidationDataFor DES	(constructor of the class)	

5.13.1 Properties

decimalTable Property (R/W)

Type	***byte[ ]***
Description	ASCII decimalization table (16 character string containing ‘0’ to ‘9’). Used to convert the hexadecimal digits (0x0 to 0xF) of the encrypted validation data to decimal digits (0x0 to 0x9).

maxPIN Property (R/W)

Type	***int***
Description	Maximum number of PIN digits to be used for validation.

noLeadingZero Property (R/W)

Type	***boolean***
Description	If set to TRUE and the first digit of result of the modulo 10 addition is a X’0’, it is replaced with X’1’ before performing the verification against the entered PIN. If set to FALSE, a leading zero is allowed in entered PINs.

offset Property (R/W)

Type	***byte [ ]***
Description	Offset for the PIN block. If this property is set to **null**, the offset is to be read from the card in the device. Optional property.

offsetUsed Property (R/W)

Type	***boolean***
Description	Specifies if offset is used for PIN validation.

paddingChar Property (R/W)

Type	***byte***
Description	Specifies the padding character for validation data.

validationData Property (R/W)

Type	***byte [ ]***
Description	Validation data. If this property is set to **null**, the validation data is to be read from the card in the device.

validationDigits Property (R/W)

Type	***int***
Description	Number of Validation digits to be used for validation.

5.13.2 Methods

JxfsPINValidationDataForDES Constructor

Syntax	***JxfsPINValidationDataForDES (java.lang.String keyName, byte[ ] keyEncrKey, byte[ ] decimalTable, int maxPIN, boolean noLeadingZero, byte[ ] offset, boolean offsetUsed, byte paddingChar, byte[ ] validationData, int validationDigits)*** ***JxfsPINValidationDataForDES (java.lang.String keyName, byte[ ] keyEncrKey, int validationTrackNumber, int validationLength, int validationIndex, int offsetTrackNumber, int offsetLength, int offsetIndex, boolean ejectCurrent, ejectWhenComplete, byte[ ] decimalTable, int maxPIN, boolean noLeadingZero, byte paddingChar, byte[ ] validationData, int validationDigits)***
Description	Constructors of the class.

5.13.3 Exceptions

Exception JXFS_E_PARAMETER_INVALID is thrown by the setter methods in the following cases:

- The value for an int property is negative.
- The value for decimal Table is null.

5.14 JxfsPINValidationDataForEC

Class that contains data required for EUROCHEQUE PIN validation.

Summary

Implements : **Extends :** *JxfsPINValidationData*

Property	Type	Access	Initialized after
DecimalTable	byte[ ]	R/W	
EurochequeData	byte[ ]	R/W	
FirstEncDigits	int	R/W	
FirstEncOffset	int	R/W	
PINVV	byte [ ]	R/W	
PINVVDigits	int	R/W	
PINVVOffset	int	R/W	

Method	Return	May use after
Get*Property*	*Property*	
Set*Property*	*void*	
JxfsPINValidationDataForEC	(constructor of the class)	

5.14.1 Properties

decimalTable Property (R/W)

Type *byte[ ]*
Description ASCII decimalization table (16 character string containing '0' to '9'). Used to convert the hexadecimal digits (0x0 to 0xF) of the encrypted validation data to decimal digits (0x0 to 0x9).

eurochequeData Property (R/W)

Type *byte[ ]*
Description Track 3 Eurocheque data.

firstEncDigits Property (R/W)

Type *int*
Description Number of digits to extract after first encryption.

firstEncOffset Property (R/W)

Type *int*
Description Offset of digits to extract after first encryption.

PINVV Property (R/W)

Type *byte [ ]*
Description PIN Validation Value from track data.

PINVVDigits Property (R/W)

Type	*int*
Description	Number of validation digits to extract for PVV.

PINVVOffset Property (R/W)

Type	*int*
Description	Offset of digits to extract for PVV.

5.14.2 Methods

JxfsPINValidationDataForEC Constructor

Syntax	***JxfsPINValidationDataForEC (java.lang.String keyName, byte[ ] keyEncrKey, byte[ ] decimalTable, byte[ ] eurochequeData, int firstEncDigits, int firstEncOffset, byte[ ] PINVV, int PINVVDigits, int PINVVOffset)***
Description	Constructor of the class.

5.14.3 Exceptions

Exception JXFS_E_PARAMETER_INVALID is thrown by the setter methods in the following cases:

- The value for an int property is negative.
- The value for decimalTable, eurochequeData or PINVV is null.

5.15 JxfsPINValidationDataForVISA

Class that contains data required for VISA PIN validation.

Summary

Implements : **Extends :** *JxfsPINValidationData*

Property	**Type**	**Access**	**Initialized after**
PAN	byte[ ]	R/W	
PINVV	byte[ ]	R/W	
PINVVDigits	int	R/W	

Method	**Return**	**May use after**
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINValidationDataFor VISA	(constructor of the class)	

5.15.1 Properties

PAN Property (R/W)

Type	***byte[ ]***
Description	Primary Account Number from track data.

PINVV Property (R/W)

Type	***byte[ ]***
Description	PIN Validation Value from track data.

PINVVDigits Property (R/W)

Type	***int***
Description	Number of digits of PVV.

5.15.2 Methods

JxfsPINValidationDataForVISA Constructor

Syntax	***JxfsPINValidationDataForVISA (java.lang.String keyName, byte[ ] keyEncrKey, byte[ ] PAN, byte[ ] PINVV, byte[ ] PINVVDigits)***
Description	Constructor of the class.

5.15.3 Exceptions

Exception JXFS_E_PARAMETER_INVALID is thrown by the setter methods in the following cases:

- The value for PINVVDigits is negative or zero.
- The value for PAN or PINVV is null.

5.16 JxfsPINOffsetData

Data class for data required for createOffset() method of JxfsSecurePINKeypad.

Summary

Implements : **Extends :** *JxfsPINValidationData*

Property	Type	Access	Initialized after
decimalTable	byte[ ]	R/W	
maxPIN	int	R/W	
paddingChar	byte	R/W	
validationData	byte[ ]	R/W	
validationDigits	int	R/W	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINOffsetData	(constructor of the class)	

5.16.1 Properties

decimalTable Property (R/W)

Type	***byte[ ]***
Description	ASCII decimalization table (16 position byte array containing ‘0’ to ‘9’ characters). Used to convert the hexadecimal digits (0x0 to 0xF) of the encrypted validation data to decimal digits (0x0 to 0x9).

maxPIN Property (R/W)

Type	***int***
Description	Maximum number of PIN digits to be used for validation.

paddingChar Property (R/W)

Type	***byte***
Description	Specifies the padding character for validation data.

validationData Property (R/W)

Type	***byte[ ]***
Description	Validation data. If this property is set to **null**, the validation data is to be read from the card in the device.

validationDigits Property (R/W)

Type	***int***
Description	Number of Validation digits to be used for validation.

5.16.2 Methods

JxfsPINOffsetData Constructor

Syntax	***JxfsPINOffsetData (java.lang.String keyName, byte[ ] keyEncrKey, byte[ ] decimalTable, int maxPIN, byte paddingChar, byte[ ] validationData, int validationDigits)*** ***JxfsPINOffsetData (java.lang.String keyName, byte[ ] keyEncrKey, int validationTrackNumber, int validationLength, int validationIndex, boolean ejectCurrent, ejectWhenComplete, byte[ ] decimalTable, int maxPIN, byte paddingChar, int validationDigits)***
Description	Constructor of the class.

5.16.3 Exceptions

Exception JXFS_E_PARAMETER_INVALID is thrown by the setter methods in the following cases:

- The value for maxPIN or validationDigits is negative or zero.
- The value for decimalTable is null.

5.17 JxfsPINBlockData

Data class for data required for *pinBlock()* method of JxfsSecurePINKeypad.

Summary

Implements : **Extends :** *JxfsPINValidationData*

Property	Type	Access	Initialized after
customerData	byte[ ]	R/W	
paddingChar	byte	R/W	
pinBlockFormat	int	R/W	
XORData	byte[ ]	R/W	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINBlockData	(constructor of the class)	

5.17.1 Properties

customerData Property (R/W)

Type	***byte[ ]***
Description	Used for ANSI, ISO-0 and ISO-1 algorithm to build the formatted PIN. For ANSI and ISO-0 the PAN (Primary Account Number) is used, for ISO-1 a ten digit transaction field is required. If not used a **null** is required. Used for DIEBOLD with coordination number, as a two digit coordination number. If this property is set to **null**, the validation data is to be read from the card in the device. Used for EMV, with the unpredictable number (8 bytes) obtained from the chip card. This number is formatted unpacked. For example if the unpredictable number is "0x01 0x23 0x45 0x67 0x89 0xAB 0xCD 0xEF", it is passed as follows "0x30 0x31 0x32 0x33 0x34 0x35 0x36 0x37 0x38 0x39 0x41 0x42 0x43 0x44 0x45 0x46" If this property is set to **null**, the validation data is to be read from the card in the device.

paddingChar Property (R/W)

Type	***byte***
Description	Specifies the padding character.

pinBlockFormat Property (R/W)

Type	***int***
Description	Specifies the format of the PIN block. Possible values are:

Value	**Meaning**
JXFS_PIN_FMT_3624	Format 3624.
JXFS_PIN_FMT_ANSI	Format ANSI.
JXFS_PIN_FMT_ISO0	Format ISO0.
JXFS_PIN_FMT_ISO1	Format ISO1.

JXFS_PIN_FMT_EC12	Format EC12.
JXFS_PIN_FMT_EC13	Format EC13.
JXFS_PIN_FMT_EC13RAND	Format EC13, random padding.
JXFS_PIN_FMT_VISA	Format VISA.
JXFS_PIN_FMT_DIEBOLD	Format DIEBOLD.
JXFS_PIN_FMT_DIEBOLDC0	Format DIEBOLD C0.
JXFS_PIN_FMT_EMV	Format EMV (RSA)

XORData Property (R/W)

Type ***byte[ ]***

Description If the formatted PIN is encrypted twice to build the resulting PIN block, this data can be used to modify the result of the first encryption by an XOR-operation.

5.17.2 Methods

JxfsPINBlockData Constructor

Syntax ***JxfsPINBlocktData (java.lang.String keyName, byte[ ] keyEncrKey, byte[ ] customerData, byte paddingChar, int pinBlockFormat, byte[ ] XORData)***

JxfsPINBlocktData (java.lang.String keyName, byte[ ] keyEncrKey, int validationTrackNumber, int validationLength, int validationIndex,, boolean ejectCurrent, ejectWhenComplete, byte paddingChar, int pinBlockFormat, byte[ ] XORData)

Description Constructor of the class.
If KeyName specifies a RSA Key, RSA encryption will be performed.

5.17.3 Exceptions

Exception JXFS_E_PARAMETER_INVALID is thrown by the setter methods in the following cases:

- The value for pinBlockFormat is out of range.
- The value for XORData is null.

5.18 JxfsPINChipValidationData

Abstract class.

The J/XFS PIN Chip Validation Data is the root of a hierarchy of data objects that contain data for PIN chip verification and used in *validationPINChip()* method of JxfsSecurePINKeypad Device Control class.

Summary

Implements : **Extends : JxfsType**

Property	Type	Access	Initialized after
presentationMode	int	R/W	
chipProtocol	int	R/W	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	

5.18.1 Properties

presentationMode Property (R/W)

Type ***int***
Description Presentation mode for which this object is intended to be used.

Set by the constructor of each of the specific subclasses of *JxfsPINChipValidationData*.

Possible values are:

Value	Meaning
JXFS_PIN_PRES_CLEAR	Clear text presentation of PIN to chip card device.

chipProtocol Property (R/W)

Type ***int***
Description Protocol to be used with chip.
Possible values are:

Value	Meaning
0 .. 15	Protocols T=0 .. T=15.

5.18.2 Exceptions

Exception JXFS_E_PARAMETER_INVALID is thrown by the setter methods in the following cases:

- The value for presentationMode or chipProtocol is out of range.

5.19 JxfsPINChipValidationDataClear

Class that contains data required for Clear chip PIN validation.

Summary

Implements :	**Extends : JxfsPINChipValidationData**

Property	Type	Access	Initialized after
chipData	byte[ ]	R/W	
insertPosition	int	R/W	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINChipValidationDataClear	(constructor of the class)	

5.19.1 Properties

chipData Property (R/W)

Type	***byte[ ]***
Description	Data to be sent to the chip.

insertPosition Property (R/W)

Type	***int***
Description	Contains the bit position where to insert the PIN in the *chipData* buffer (0 means is bit 0 of first byte, and so on).

5.19.2 Methods

JxfsPINChipValidationDataClear Constructor

Syntax	***JxfsPINChipValidationDataClear (int chipProtocol, byte[ ] chipData, int insertPosition)***
Description	Constructor of the class.

5.19.3 Exceptions

Exception JXFS_E_PARAMETER_INVALID is thrown by the setter methods in the following cases:

- The value for insertPosition is negative.
- The value for chipData is null.

5.20 JxfsPINValidationResult

This class contains the result of a PIN validation operation.

Summary

Implements :		**Extends :**	**JxfsType**

Property	Type	Access	Initialized after
validationResult	boolean	R	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINValidationResult	(constructor of the class)	

5.20.1 Properties

validationResult Property (R)

Type	***boolean***
Description	TRUE if PIN was validated, otherwise FALSE.

5.20.2 Methods

JxfsPINValidationResult Constructor

Syntax	***JxfsPINValidationResult (boolean validationResult)***
Description	Constructor of the class.

5.21 JxfsPINOffset

This class contains a PIN offset.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
offsetValue	byte[ ]	R	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINOffset	(constructor of the class)	

5.21.1 Properties

offsetValue Property (R)

Type	***byte[ ]***
Description	A PIN Offset

5.21.2 Methods

JxfsPINOffset Constructor

Syntax ***JxfsPINOffset (byte[ ] offsetValue)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	*offsetValue* is null.

5.22 JxfsPINBlock

This class contains a PIN block.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
PINBlockValue	byte[ ]	R	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINBlock	(constructor of the class)	

5.22.1 Properties

PINBlockValue Property (R)

Type	***byte[ ]***
Description	A PIN Block.

5.22.2 Methods

JxfsPINBlock Constructor

Syntax	***JxfsPINBlock (byte[ ] PINBlockValue)***
Description	Constructor of the class.

5.23 JxfsPINChipValidationResult

This class contains the result of a PIN chip validation operation.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
validationResult	byte[ ]	R	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINChipValidationResult	(constructor of the class)	

5.23.1 Properties

validationResult Property (R)

Type	***byte[ ]***
Description	Data returned from chip.

5.23.2 Methods

JxfsPINChipValidationResult Constructor

Syntax ***JxfsPINChipValidationResult (byte[ ] validationResult)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	Meaning
JXFS_E_PARAMETER_INVALID	*validationResult* is null.

5.24 JxfsPINCryptoModes

This class provides properties and methods to query which encryption modes are supported by a secure PIN device service.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
cryptDESECB	boolean	R	
cryptDESCBC	boolean	R	
cryptDESCFB	boolean	R	
cryptDESMAC	boolean	R	
cryptRSA	boolean	R	
cryptECMA	boolean	R	
cryptTRIDESECB	boolean	R	
cryptTRIDESCBC	boolean	R	
cryptTRIDESCFB	boolean	R	
cryptTRIDESMAC	boolean	R	

Method	Return	May use after
is*Property*	*Property*	
JxfsPINCryptoModes	(constructor of the class)	

5.24.1 Properties

cryptDESECB Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports Electronic Code Book encryption.

Value	Meaning
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

cryptDESCBC Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports Cipher Block Chaining encryption.

Value	Meaning
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

cryptDESCFB Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports Cipher Feed Back encryption.

Value	Meaning
FALSE	Encryption mode is not supported.

TRUE	Encryption mode is supported.

cryptDESMAC Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports MAC calculation using CBC.

Value	**Meaning**
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

cryptRSA Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports RSA encryption.

Value	**Meaning**
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

cryptECMA Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports ECMA encryption.

Value	**Meaning**
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

cryptTRIDESECB Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports Triple DES with Electronic Code Book.

Value	**Meaning**
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

cryptTRIDESCBC Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports Triple DES with Cypher Block Chaining.

Value	**Meaning**
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

cryptTRIDESCFB Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports Triple DES with Cipher Feed Back.

Value	**Meaning**
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

cryptTRIDESMAC Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports Triple DES MAC calculation using CBC .

Value	**Meaning**
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

5.24.2 Methods

JxfsPINCryptoModes Constructor

Syntax	***JxfsPINCryptoModes (boolean cryptDESECB, boolean cryptDESCBC, boolean cryptDESCFB, boolean cryptDESMAC, boolean cryptRSA, boolean cryptECMA, boolean cryptTRIDESECB, boolean cryptTRIDESCBC, boolean cryptTRIDESCFB, boolean cryptTRIDESMAC)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	All the parameters are false.

5.25 JxfsPINEMVCryptoModes

This class provides properties and methods to query which encryption modes are supported by a secure PIN device service for importing a RSA public key for EMV.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
EMVPlainTextCA	boolean	R	
EMVChecksumCA	boolean	R	
EMVEPICA	boolean	R	
EMVIssuer	boolean	R	
EMVICC	boolean	R	
EMVICCPIN	boolean	R	
EMVPKCSV1_5CA	boolean	R	
Hash_SHA1	boolean	R	

Method	Return	May use after
is*Property*	*Property*	
JxfsPINEMVCryptoModes	(constructor of the class)	

5.25.1 Properties

EMVPlainTextCA Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports the import of a Certification Authority plain text CA public key with no verification.

Value	Meaning
FALSE	Plain text mode not supported.
TRUE	Plain text mode is supported.

EMVChecksumCA Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports the import of a Certification Authority plain text public key using the EMV 2000 verification algorithm.

Value	Meaning
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

EMVEPICA Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports the import of a Certification Authority public key using the self sign scheme defined in the EUROPAY International, EPI CA module Technical- Interface specification Version 1.4

Value **Meaning**

FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

EMVIssuer Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports the import of an issuer public key as defined in the EMV 2000 book II.

Value	**Meaning**
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

EMVICC Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports the import of an ICC public key as defined in EMV specifications book II.

Value	**Meaning**
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

EMVICCPIN Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports the import of an ICC PIN public key as defined in EMV specifications book II.

Value	**Meaning**
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

EMVPKCSV1_5CA Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports the import of a certification Authority public key verified using a signature generated with a private key for which the public key is already loaded.

Value	**Meaning**
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

Hash_SHA1 Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports SHA1 digest algorithm.

Value	**Meaning**
FALSE	Encryption mode is not supported.
TRUE	Encryption mode is supported.

5.25.2 Methods

JxfsPINEMVCryptoModes Constructor

Syntax	***JxfsPINEMVCryptoModes (boolean EMVPlainTextCA, boolean EMVChecksumCA, boolean EMVEPICA, boolean EMVIssuer, boolean EMVICC, boolean EMVICCPIN, boolean EMVPKCSV1_5CA, boolean Hash_SHA1)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	All the parameters are false.

5.26 JxfsPINKeyDetail

The J/XFS PIN Key Detail data class contains relevant information for an application about a key in the device's key table.

Summary

Implements : **Extends : JxfsType**

Property	Type	Access	Initialized after
keyLoaded	boolean	R	
keyName	java.lang.String	R	
keyReload	boolean	R	
keyUse	JxfsPINKeyUses	R	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINKeyDetail	(constructor of the class)	

5.26.1 Properties

keyLoaded Property (R)

Type ***boolean***
Description Indicates whether the key has been loaded/imported.

Value	**Meaning**
TRUE	Key has been loaded/imported and is ready to be used.
FALSE	Key is not operationally ready.

keyName Property (R)

Type ***jave.lang.String***
Description Name of the key.

keyReload Property (R)

Type ***boolean***
Description Indicates whether the key can be loaded/imported just once.

Value	**Meaning**
TRUE	Key can be loaded/imported.
FALSE	Key can only be loaded/imported once.

keyUse Property (R)

Type ***JxfsPINKeyUses***
Description Type of access for which the key is intended to be used.

5.26.2 Methods

JxfsPINKeyDetail Constructor

Syntax	***JxfsPINKeyDetail (boolean keyLoaded, java.lang.String keyName, boolean keyReload, JxfsKeyUses keyUse)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *keyName* is null. *keyUse* is null.

5.27 JxfsPINKeyToImport

The J/XFS PIN Key to Import data class contains data required as input for *importKey()* operation.

Summary

Implements : **Extends : JxfsType**

Property	Type	Access	Initialized after
Key	java.lang.String	R/W	
keyEncKey	java.lang.String	R/W	
keyReload	boolean	R/W	
keyUse	JxfsPINKeyUses	R/W	
keyValue	byte[ ]	R/W	
idKey	byte[ ]	R/W	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINKeyToImport	(constructor of the class)	

5.27.1 Properties

key Property (R/W)

Type	***java.lang.String***
Description	Name of the key being loaded.

keyEncKey Property (R/W)

Type	***java.lang.String***
Description	Name of the key encrypting key that was used to encrypt the *keyValue* property data. If this property is set to null, the key specified in *keyValue* is directely stored in the device‘s key table.

keyReload Property (R/W)

Type	***boolean***
Description	Indicates whether the key can be loaded only once.

Value	**Meaning**
TRUE	Key can be loaded/imported may times.
FALSE	Key can only be loaded/imported once.

keyUse Property (R/W)

Type	***JxfsPINKeyUses***
Description	Type of access for which the key is intended to be used.

keyValue Property (R/W)

Type	***byte[ ]***
Description	Key value.

idKey Property (R/W)

Type	***byte[ ]***

Description	Specifies the key owner identification or null.

5.27.2 Methods

JxfsPINKeyToImport Constructor

Syntax ***JxfsPINKeyToImport (java.lang.String key, java.lang.String keyEncKey, boolean keyReload, JxfsKeyUses keyUse, byte[ ] keyValue, byte[ ] idKey)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *key* is null. *keyUse* is null. *keyValue* is null. *idKey* is null.

5.28 JxfsPINEMVRSAKeyToImport

The JxfsPINEMVRSAKeyToImportdata class contains data required as input for *importEMVRSAKey()* operation. This class is similar to JxfsPINKeyToImport but it is specifically designed to address the key formats and security features defined by EMV.

This class is used to import the EMV RSA public keys. The RSA keys to import are provided either by the Certification Authority (VISA or MASTERCARD EUROPE), or by the EMV application in the chip card (ISSUER KEY and ICC KEY).

Summary

Implements : **Extends : JxfsType**

Property	Type	Access	Initialized after
keyName	java.lang.String	R/W	
keyUse	JxfsPINKeyUses	R/W	
idKey	byte[ ]	R/W	
EMVRSAIntegrityAlgorithm	JxfsPINEMVRSAIntegrityAlgorithm	R/W	
EMVRSAIntegrityData	byte []	R/W	
signatureKeyName	java.lang.String	R/W	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINEMVRSAKeyToImport	(constructor of the class)	

5.28.1 Properties

keyName Property (R/W)

Type	***java.lang.String***
Description	Name of the key being loaded.

keyUse Property (R/W)

Type	***JxfsPINKeyUses***
Description	Type of access for which the key is intended to be used. Only the properties *kuseRSAPublicEncrypt* and *kuseRSAPublicVerify* are supported

idKey Property (R/W)

Type	***byte[ ]***
Description	Specifies the key owner identification or null.

EMVRSAIntegrityAlgorithm Property (R/W)

Type	***jxfsPINEMVRSAIntegrityAlgorithm***
Description	Specifies the algorithm used to verify the integrity of the Certification Authority RSA public key. See JxfsPINEMVRSAIntegrity data class for more detailed information

EMVRSAIntegrityData Property (R/W)

Type	***byte[ ]***
Description	Contains all the necessary data to complete the import RSA public key according to the EMVRSAIntegrityAlgorithm property. The content of this parameter in dependant of the EMVRSAIntegrityAlgorithm parameter:

plaintext_CA: EMVRSAIntegrityData contains a DER encoded PKCS#1 public key. No verification is possible. signatureKeyName is ignored.
checksum_CA : EMVRSAIntegrityData contains table 23 data, as specified in EMV 2000 Book 2.The plain text key is verified as defined within EMV2000 Book 2.signatureKeyName is ignored.
EPI_CA : EMVRSAIntegrityData contains the concatenation of tables 4 and 13, as specified in " Europay International, EPI CA Module Technical – Interface specification Version 1.4. These tables are also described in the EMV Clarifications Appendix
signatureKeyName is ignored.
issuer : EMVRSAIntegrityData contains the EMV public key certificate. It consists of the concatenation of :

- the key exponent length (1 byte),
- the key exponent value (variable length – EMV Tag value : ‘9F32’),
- the EMV certificate length (1 byte), the EMV certificate value (variable length – EMV Tag value : ‘90’) ,
- the remainder length (1 byte).
- The remainder value (variable length – EMV Tag value : ‘92’),
- the PAN length (1 byte)
- and the PAN value (variable length – EMV Tag value : ‘5A’).

The device services will compare the leftmost three-eight digits of the PAN to the Issuer Identification Number retrieved from the certificate. For more explanations, the reader can refer to EMVco, Book2 – Security & Key Management Version 4.0, Table 4.
signatureKeyName defines the previously loaded key used to verify the signature
ICC : EMVRSAIntegrityData contains the EMV public key certificate. It consists of the concatenation of :

- the key exponent length (1 byte),
- the key exponent value (variable length– EMV Tag value : ‘9F47’),
- the EMV certificate length (1 byte),
- the EMV certificate value (variable length – EMV Tag value :’9F46’),
- the remainder length (1 byte),
- the remainder value (variable length – EMV Tag value : ‘9F48’),
- the SDA length (1 byte), the SDA value (variable length),
- the PAN length (1 byte)
- and the PAN value (variable length – EMV Tag value : ‘5A’),

The Device Services will compare the PAN to the PAN retrieved from the certificate. For more explanations, the reader can refer to EMVco, Book2 – Security & Key Management Version 4.0, Table 9.
signatureKeyName defines the previously loaded key used to verify the signature
ICC_PIN : EMVRSAIntegrityData contains the EMV public key certificate. It consists of the concatenation of :

- the key exponent length (1 byte),
- the key exponent value (variable length – EMV Tag value : ‘9F2E’),
- the EMV certificate length (1 byte),
- the EMV certificate value (variable length – EMV Tag value :’9F2D’),
- the remainder length (1 byte),
- the remainder value (variable length – EMV Tag value : ‘9F2F’),
- the SDA length (1 byte),
- the SDA value (variable length),
- the PAN length (1 byte)
- and the PAN value (variable length – EMV Tag value : ‘5A’).

The Device services will compare the PAN to the PAN retrieved from the certificate. For more explanations, the reader can refer to EMVco,

Book2 – Security & Key Management Version 4.0, Table 9. signatureKeyName defines the previously loaded key used to verify the signature
PKCSV1_5_CA : EMVRSAIntegrityData contains the CA public key signed with the previously loaded public key specified in signatureKeyName. lpxImportData consists of the concatenation of EMV 2000 Book II Table 23) + 8 byte random number + Signature. The 8 byte random number is not used for validation; it is used to ensure the signature is unique. The Signature consists of all the bytes in the EMVRSAIntegrityData buffer after table 23 and the 8 byte random number

signatureKeyName Property (R/W)

Type	***java.lang.String***
Description	Specifies the name of an asymmetric key, previously stored, which will be used to compute the certificate defined in JxfsPINEMVRSAIntegrity .

5.28.2 Methods

JxfsPINEMVKeyToImport Constructor

Syntax ***JxfsEMVPINKeyToImport (java.lang.String keyName, JxfsPINKeyUses keyUse, byte[ ] idKey, JxfsPINEMVRSAIntegrityAlgorithm EMVRSAIntegrityAlgorithm, byte [ ] EMVRSAIntegrityData, java.lang.String signatureKeyName)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *keyName* is null or not set correctly. *keyUse* is null or not set correctly. *idKey* is null. *EMVRSAIntegrityAlgorithm* is null or not set correctly *signatureKeyName* is null or not set correctly

5.29 JxfsPINInitialization

This class contains the result of a security module's initialization operation.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
idKey	byte[ ]	R	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINInitialization	(constructor of the class)	

5.29.1 Properties

idKey Property (R)

Type	***byte[ ]***
Description	Value of the ID key encrypted by the ID encryption key. Can be used as authorization for importKey() method. Null if not supported by the device.

5.29.2 Methods

JxfsPINInitialization Constructor

Syntax	***JxfsPINInitialization (byte[ ] idKey)***
Description	Constructor of the class.

5.30 JxfsPINKeyVerificationData

This class contains data returned after the completion of a *importKey()* operation..

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
keyVerCode	byte[ ]	R	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINKeyVerificationData	(constructor of the class)	

5.30.1 Properties

keyVerCode Property (R)

Type	***byte[ ]***
Description	Key verification code data that can be used for verification of the loaded key. For the importEMVRSAPublicKey , if applied , it contains the expiry date of the certificate in the following format YYYY-MM Null if this function is not supported by the device.

5.30.2 Methods

JxfsPINKeyVerificationData Constructor

Syntax	***JxfsPINKeyVerificationData (byte[ ] keyVerCode)***
Description	Constructor of the class.

5.31 JxfsPINCryptoData

The J/XFS PIN Cryptographic data class contains data required for encryption/decryption methods.

Summary

Implements : **Extends : JxfsType**

Property	Type	Access	Initialized after
cryptoMode	int	R/W	
data	byte[ ]	R/W	
key	java.lang.String	R/W	
keyEncKey	byte[]	R/W	
paddingChar	byte	R/W	
startValue	byte[ ]	R/W	
startValueKey	java.lang.String	R/W	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINCryptoData	(constructor of the class)	

5.31.1 Properties

cryptoMode Property (R/W)

Type ***Int***

Description Indicates the algorithm to be used.

Value	**Meaning**
JXFS_PIN_CRYPT_MODE_DESECB	Electronic Code Book
JXFS_PIN_CRYPT_MODE_DESCBC	Cipher Block Chaining
JXFS_PIN_CRYPT_MODE_DESMAC	MAC calculation using CBC
JXFS_PIN_CRYPT_MODE_DESCFB	Cipher Feed Back
JXFS_PIN_CRYPT_MODE_RSA	RSA Encryption
JXFS_PIN_CRYPT_MODE_ECMA	ECMA Encryption
JXFS_PIN_CRYPT_MODE_TRIDESECB	Triple DES with Electronic Code Book
JXFS_PIN_CRYPT_MODE_TRIDESCBC	Triple DES with Cipher Block Chaining
JXFS_PIN_CRYPT_MODE_TRIDESCFB	Triple DES with Cipher Feed Back
JXFS_PIN_CRYPT_MODE_TRIDESMAC	Triple DES MAC calculation using CBC

data Property (R/W)

Type ***byte[ ]***

Description Data to be encrypted, decrypted or MACed.

key Property (R/W)

Type	***java.lang.String***
Description	Name of the key to be used in cryptographic operation.

keyEncKey Property (R/W)

Type	***byte[]***
Description	Encrypted key, under the key contained in *key* property, to be used in cryptographic operation. If null, key contained in *key* property is used.

paddingChar Property (R/W)

Type	***byte***
Description	Specifies the padding character used.

startValue Property (R/W)

Type	***byte[ ]***
Description	DES and Triple DES initialization vector for the CBC, CFB and MAC. If null, s*tartValueKey* property is used as the Initialization Vector. If both are null the default is 16 hexadecimal digits 0x00.

startValueKey Property (R/W)

Type	***java.lang.String***
Description	Name of the stored key used to decrypt the s*tartValue* property to obtain the Initialization Vector. If null, *startValue* is used as the initialization vector.

5.31.2 Methods

JxfsPINCryptoData Constructor

Syntax	***JxfsPINCryptoData (int cryptoMode, byte[ ] data, java.lang.String key, java.lang.String keyEncKey, byte paddingChar, byte[ ] startValue, java.lang.String startValueKey)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	Meaning

	JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *cryptoMode* is out of range. *data* is null. *key* is null or not set correctly *keyEncKey* is null or not set correctly *paddingChar* is null *startValue* is null or not set correctly *startValueKey* is null or not set correctly.

5.32 JxfsPINMACData

The J/XFS PIN Cryptographic MAC data class contains data required for MAC generation operation.
It is a subclass of *JxfsPINCryptoData.*

Summary

Implements : **Extends : JxfsPINCryptoData**

Property	Type	Access	Initialized after
compression	boolean	R/W	
compressionChar	byte	R/W	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINMACData	(constructor of the class)	

5.32.1 Properties

compression Property (R/W)

Type	***boolean***
Description	Specifies whether data is to be compressed (blanks removed) before building the MAC.

compressionChar Property (R/W)

Type	***byte***
Description	If compression is **TRUE**, it specifies the representation of the blank character in the actual code table.

5.32.2 Methods

JxfsPINMACData Constructor

Syntax	***JxfsPINMACData (boolean compression, byte compressionChar)***
Description	Constructor of the class.

5.33 JxfsPINCryptoResult

The J/XFS PIN Cryptographic result data class contains data returned by cryptographic operations (encrypt, decrypt and generateMAC).

Summary

Implements : **Extends : JxfsType**

Property	Type	Access	Initialized after
cryptoResult	byte[ ]	R	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINCryptoResult	(constructor of the class)	

5.33.1 Properties

cryptoResult Property (R/W)

Type	***byte[ ]***
Description	Data returned by a cryptographic operation.

5.33.2 Methods

JxfsPINCryptoResult Constructor

Syntax ***JxfsPINCryptoResult (byte[ ] cryptoResult)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	*cryptoResult* is null.

5.34 JxfsPINKeyUses

Summary

This class provides properties and methods to query which type of access a key is intended for.

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
kuseEncDec	boolean	R/W	
kusePin	boolean	R/W	
kuseMac	boolean	R/W	
kuseKek	boolean	R/W	
kuseVek	boolean	R/W	
kuseMaster	boolean	R/W	
kuseRSAPublicEncrypt	boolean	R/W	
kuseRSAPublicVerify	boolean	R/W	
kuseRSAPrivateSign	boolean	R/W	
kuseRSAPrivate	boolean	R/W	

Method	Return	May use after
is*Property*	*Property*	
set*Property*	*void*	
JxfsPINKeyUses	(constructor of the class)	

5.34.1 Properties

kuseEncDec Property (R/W)

Type	***boolean***	
Description	Indicates if the key may be used for encryption and decryption.	
	Value	**Meaning**
	FALSE	This use is not supported.
	TRUE	This use is supported.

kusePin Property (R/W)

Type	***boolean***	
Description	Indicates if the key may be used for PIN functions.	
	Value	**Meaning**
	FALSE	This use is not supported.
	TRUE	This use is supported.

kuseMac Property (R/W)

Type	***boolean***	
Description	Indicates if the key may be used for MAC generation.	
	Value	**Meaning**
	FALSE	This use is not supported.
	TRUE	This use is supported.

kuseKek Property (R/W)

Type	***boolean***	
Description	Indicates if the key may be used as key encryption key.	
	Value	**Meaning**

FALSE	This use is not supported.
TRUE	This use is supported.

kuseVek Property (R/W)

Type *boolean*

Description Indicates if the key may be used as CBC Start Value encryption key.

Value	**Meaning**
FALSE	This use is not supported.
TRUE	This use is supported.

kuseMaster Property (R/W)

Type *boolean*

Description Indicates if the key may be used as Master encryption key.

Value	**Meaning**
FALSE	This use is not supported.
TRUE	This use is supported.

kuseRSAPublicEncrypt Property (R/W)

Type *boolean*

Description Indicates if the key may be used as Public key for RSA encryption or for EMV PIN Block creation.

Value	**Meaning**
FALSE	This use is not supported.
TRUE	This use is supported.

kuseRSAPublicVerify Property (R/W)

Type *boolean*

Description Indicates if the key may be used as a public key for RSA verification

Value	**Meaning**
FALSE	This use is not supported.
TRUE	This use is supported.

kuseRSAPrivate Property (R/W)

Type *boolean*

Description Indicates if the key may be used as a private key for RSA encryption

Value	**Meaning**
FALSE	This use is not supported.
TRUE	This use is supported.

kuseRSAPrivateSign Property (R/W)

Type *boolean*

Description Indicates if the key may be used as a private key for RSA signature generation RSA encryption

Value	**Meaning**
FALSE	This use is not supported.
TRUE	This use is supported.

5.34.2 Methods

JxfsPINKeyUses Constructor

Syntax ***JxfsPINKeyUses (boolean kuseEncDec, boolean kusePin, boolean kuseMac, boolean kuseKek, boolean kuseVek, boolean kuseMaster)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	All the parameters are false.

JxfsPINKeyUses Constructor

Syntax ***JxfsPINKeyUses (boolean kuseEncDec, boolean kusePin, boolean kuseMac, boolean kuseKek, boolean kuseVek, boolean kuseMaster, boolean kuseRSAPublicEncrypt, boolean kuseRSAPublicVerify, boolean kuseRSAPrivate, boolean kuseRSAPrivateSign)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	All the parameters are false.

5.35 JxfsPINIdKeyModes

This class provides properties and methods to query which type of uses of ID keys are implemented.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
idKeyInitialize	boolean	R	
idKeyImport	boolean	R	

Method	Return	May use after
is*Property*	*Property*	
JxfsPINIdKeyModes	(constructor of the class)	

5.35.1 Properties

idKeyInitialize Property (R)

Type	***boolean***	
Description	ID key is supported in the Initialize method.	
	Value	**Meaning**
	FALSE	Feature is not supported.
	TRUE	Feature is supported.

idKeyImport Property (R)

Type	***boolean***	
Description	ID key is supported in the ImportKey method.	
	Value	**Meaning**
	FALSE	Feature is not supported.
	TRUE	Feature is supported.

5.35.2 Methods

JxfsPINIdKeyModes Constructor

Syntax	***JxfsPINIdKeyModes (boolean idKeyInitialize, boolean idKeyImport)***	
Description	Constructor of the class.	
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.	
	Value	**Meaning**
	JXFS_E_PARAMETER_INVALID	All the parameters are false.

5.36 JxfsPINEMVRSAIntegrityAlgorithm

This class provides properties and methods to query which type of verification is to be used for the verification of an RSA public key when it is loaded in the encryption module. This class is used with JxfsPINEMVRSAKeyToImport class
Only one property can be set to true.

Summary

Implements : **Extends :** **JxfsType**

Property	Type	Access	Initialized after
plaintext_CA	boolean	R/W	
checksum_CA	boolean	R/W	
EPI_CA	boolean	R/W	
issuer	boolean	R/W	
ICC	boolean	R/W	
ICC_PIN	boolean	R/W	
PKCSV1_5_CA	boolean	R/W	

Method	Return	May use after
set*Property*	*Property*	
get*Property*	*Property*	
JxfsPINEMVRSAIntegrity Algorithm	(constructor of the class)	

5.36.1 Properties

plaintext_CA Property (R/W)

Type ***boolean***

Description No integrity verification will be applied. It is used for keys provided by a Certification Authority with no verification.
The two parts of the key (modulus and exponent) are passed in clear mode as a DER encoded PKCS#1 public key. The key is loaded directly in the encryption module. This method is used by VISA

Value	Meaning
FALSE	This verification method is not applied
TRUE	This verification method is applied

checksum_CA Property (R/W)

Type ***boolean***

Description A plaintext CA public key is imported using the EMV 2000 Book II verification algorithm and it is verified before being loaded in the encryption module.
the checksum value is computed on the contents of all parts of the certification Authority public key (See EMV 2000, book 2 , p 71, table 13)

Value	Meaning
FALSE	This verification method is not applied
TRUE	This verification method is applied

EPI_CA Property (R/W)

Type ***boolean***

Description	Use of EPI CA (MASTERCARD EUROPE) Key self signed integrity verification method. This key is provided in self signed format.

Value	**Meaning**
FALSE	This verification method is not applied
TRUE	This verification method is applied

issuer Property (R/W)

Type	***boolean***
Description	An Issuer public key is imported as defined in EMV 2000 Book

Value	**Meaning**
FALSE	This verification method is not applied
TRUE	This verification method is applied

ICC Property (R/W)

Type	***boolean***
Description	An ICC public key is imported as defined in EMV 2000 Book II

Value	**Meaning**
FALSE	This verification method is not applied
TRUE	This verification method is applied

ICC_PIN Property (R/W)

Type	***boolean***
Description	An ICC PIN public key is imported as defined in EMV 2000 Book II

Value	**Meaning**
FALSE	This verification method is not applied
TRUE	This verification method is applied

PKCSV1_5_CA Property (R/W)

Type	***boolean***
Description	A Certification Authority CA public key is imported and verified using a signature generated with a private key for which the public key is already loaded

Value	**Meaning**
FALSE	This verification method is not applied
TRUE	This verification method is applied

5.36.2 Methods

JxfsPINRSAIntegrityAlgorithm Constructor

Syntax	***JxfsPIN RSAIntegrityAlgorithm (boolean plainText_CA, boolean checksum_CA ,boolean EPI_CA, boolean issuer, boolean ICC, boolean ICC_PIN, boolean PKCSV1_5_CA)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	All the parameters are false. or more then one parameter is set to true.

5.37 JxfsSHA1Data

Data class containing data

-As an input for computing a digest using a SHA_1 algorithm.

-As an output for the result of "SHA-1" algorithm

Summary

Implements : **Extends :** **JxfsType**

Property	**Type**	**Access**	**Initialized after**
length	int	R/W	
SHA1Data	byte[ ]	R/W	

Method	**Return**	**May use after**
Property	*Property*	
JxfsSHA1Data	(constructor of the class)	

5.37.1 Properties

length Property (R)

Type	***int***
Description	If it is an input parameter, it specifies the length of the data to be hashed. if it is an input parameter If it is an output parameter, it specifies the length of thehasched code. Length of the hashed code (result)

SHA1Data Property (R)

Type	***byte[ ]***
Description	Data to be hashed if it is an input parameter Digest data (result)

5.37.2 Methods

JxfsSHA1Data Constructor

Syntax	***JxfsSHA1Data (int length, byte [ ] SHA1Data)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	All the parameters are false.

5.38 JxfsPINImportRSAPublicKey

The JxfsPINImportRSAPublicKey data class contains data required as input for *importRSAPublicKey()* operation. This class is similar to *JxfsPINKeyToImport* but it is specifically designed to address the RSA public key formats.

Summary

Implements: **Extends: JxfsType**

Property	Type	Access	Initialized after
key	java.lang.String	RW	
keyUse	JxfsPINKeyUses	RW	
keyValue	byte[]	RW	
signatureKey	java.lang.String	RW	
RSASignatureAlgorithm	JxfsPINRSASignatureAlgo	RW	
signature	byte[]	RW	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINImportRSAPublicKey	(Constructor of the class)	

5.38.1 Properties

key Property (R/W)

Type	***java.lang.String***
Description	Name of the key being loaded.

keyUse Property (R/W)

Type	***JxfsPINKeyUses***
Description	Type of access for which the key is intended to be used. The valid properties for this kind of key are kuseRSAPublicEncrypt and kuseRSAPublicVerify

KeyValue Property (R/W)

Type	***byte []***
Description	Specifies the value of the public RSA key to be loaded. It is a PKCS #1 formatted RSA public key represented in DER encoded ASN.1.

signatureKey Property (R/W)

Type	***java.lang.String***
Description	Specifies the name of an asymmetric key, previously stored in the encryptor, which will be used to verify the signature passed in the signature property.

RSASignatureAlgorithm Property (R/W)

Type	***JxfsPINRSASignatureAlgo***
Description	Specifies the algorithm used to generate the signature specified in signature property.

signature Property (W)

Type	***byte [ ]***
Description	Contains the signature associate with the key being imported. The Signature is used to validate the key has been received from a trusted sender. Contains NULL when no key validation is required.

5.38.2 Methods

JxfsPINImportRSAPublicKey Constructor

Syntax	***JxfsPINImportRSAPublicKey (java.lang.String key, JxfsPINKeyUses keyUse, byte[ ] keyValue, JxfsPINRSAHashAlgorithm hashAlgorithm, byte [ ] hashData, java.lang.String signatureKey, JxfsPINRSASignatureAlgo RSASignatureAlgorithm, byte [ ] signature)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *key* is null or not set correctly . *keyUse* is null or not set correctely *keyValue* is null or not set correctly *hashAlgorithm* is null or not set correctely signaturekey is null or not set correctly. *RSASignatureAlgorithm* is null or not set correctly *signature* is null or not set correctly

5.39 JxfsPINExportRSAPublicKey

The JxfsPINExportRSAPublicKey data class contains information data that specifies the RSA public key to export

Summary

Implements: **Extends: JxfsType**

Property	Type	Access	Initialized after
key	java.lang.String	RW	Name of the key to export
keyType	JxfsPINRSAKeyType	RW	Name of the key to export

Method	Return	May use after
set*Property*	*Property*	
JxfsPINExportRSAPublic Key	(Constructor of the class)	

5.39.1 Properties

key Property (R/W)

Type	***java.lang.String***
Description	Specifies the name of the public key to be exported. This can either be the name of a key-pair generated through *generateRSAKeyPair* operation or the name of one of the default key-pairs installed during manufacture the exported RSA public key part.

keyType Property (R/R)

Type	***JxfsPINRSAKeyType***
Description	Specifies the PIN device RSA Key to export.

5.39.2 Methods

JxfsPINExportRSAPublicKey Constructor

Syntax ***JxfsPINExportRSAPublicKey (java.lang.String key, JxfsPINRSAKeyType keyType)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *key* is null. or not set correctly *keyType* is null or not set correctly

5.40 JxfsPINExportedRSAPublicKey

The JxfsPINExportedRSAPublicKey data class contains data returned on *JxfsOperationCompleteEvent* of the *exportRSAPublicKey()* operation.

Summary

Implements: **Extends: JxfsType**

Property	Type	Access	Initialized after
keyValue	byte[ ]	R	
RSASignatureAlgorithm	JxfsPINRSASignatureAlgo	R	
signature	byte[]	R	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINExportedRSAPublicKey	(Constructor of the class)	

5.40.1 Properties

KeyValue Property (R)

Type	***byte []***
Description	Contains the exported RSA public key part.

RSASignatureAlgorithm Property (R)

Type	***JxfsPINRSASignatureAlgo***
Description	Specifies the algorithm used to generate the signature specified in signature property.

signature Property (R)

Type	***byte [ ]***
Description	Contains the signature of the RSA public Key exported.

5.40.2 Methods

JxfsPINExportedRSAPublicKey Constructor

Syntax ***JxfsPINExportedRSAPublicKey (byte[ ] keyValue, JxfsPINRSAHashAlgorithm hashAlgorithm, JxfsPINRSASignatureAlgo RSASignatureAlgorithm, JxfsPINRSAKeyType signaturekey, byte [ ] signature)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Any of the following conditions is

5.41 JxfsPINImportRSADESEncipheredPublicKey

The JxfsPINImportRSADESEncipheredPublicKey data class contains data required as input for *importRSADESEncipheredPublicKey()* operation. This class is similar to JxfsPINKeyToImport but it is specifically designed to address the RSA enciphered public key formats.

Summary

Implements: **Extends: JxfsType**

Property	Type	Access	Initialized after
key	java.lang.String	R/W	
keyUse	JxfsPINKeyUses	R/W	
keyValue	byte[]	R/W	
encipherAlgorithm	JxfsPINRSASignatureAlgo	R/W	
HashAlgorithm	JxfsPINRSAHashAlgorithm	R/W	
HashData	byte []	R/W	
signatureKey	java.lang.String	R/W	
RSASignatureAlgorithm	JxfsPINRSASignatureAlgo	R/W	
signature	byte[]	R/W	
keyVerification	byte[]	R/W	

Method	Return	May use after
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINImportRSADESEncipheredPublicKey	(Constructor of the class)	

5.41.1 Properties

key Property (R/W)

Type ***java.lang.String***
Description Name of the key being loaded.

keyUse Property (R/W)

Type ***JxfsPINKeyUses***
Description Type of access for which the key is intended to be used.

KeyValue Property (R/W)

Type ***byte []***
Description Specifies the value of the public RSA key to be loaded. It contains the concatenation of the random number (when present) and enciphered key

encipherAlgorithm Property (R/W)

Type ***JxfsPINRSASignatureAlgo***
Description Specifies the RSA algorithm that is used, along with the private key of the PIN, to decipher the imported key.

hashAlgorithm Property (R/W)

Type ***JxfsPINRSAHashAlgorithms***
Description Specifies the algorithm used to generate the Hash value for the key

hashData Property (R/W)

Type ***byte [ ]***

Description	The Hash data is used to verify the key is still valid after it has been stored by the PIN. The PIN runs the stored key through the algorithm described by hashAlgorithm. The result should be identical to the value contained within hashData .

signatureKey Property (R/W)

Type	***java.lang.String***
Description	Specifies the name of an asymmetric key, previously stored, which will be used to verify the signature passed in the signature property.

RSASignatureAlgorithm Property (R/W)

Type	***JxfsPINRSASignatureAlgo***
Description	Specifies the algorithm used to generate the signature specified in signature property.

signature Property (R/W)

Type	***byte [ ]***
Description	Contains the signature associate with the key being imported. The Signature is used to validate the key has been received from a trusted sender. Contains NULL when no key validation is required.

keyVerification Property (R/W)

Type	***byte [ ]***
Description	Contains the key verification code data that can be used for verification of the loaded key, NULL if device does not have that capability

5.41.2 Methods

JxfsPINImportRSADESEncipheredPublicKey Constructor

Syntax	***JxfsPINImportRSADESEncipheredPublicKey (java.lang.String key, JxfsPINKeyUses keyUse, byte[ ] keyValue, byte[ ] keyAttribute, JxfsPINRSASignatureAlgo encipherAlgorithm, java.util.vector controlVector, JxfsPINRSAHashAlgorithm hashalgorithm, byte [ ]hashData, java.lang.String signatureKey, JxfsPINRSASignatureAlgo RSASignatureAlgorithm, byte [ ] signature, byte [ ] keyVerification)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**

JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *key* is null or not set correctly. *keyUse* is null or not set correctly. *keyValue* is null or not set correctly *keyAttribute* is null or not set correctly. *encipherAlgorithm* is null or not set correctly. *hashAlgorithm* is null or not set correctly. *hashData* is null. *signatureKey* is null or not set correctly. *RSAsignatureAlgorithm* is null or not set correctly. *signature* is null or not set correctly. *keyverification* is null or not set correctly.

5.42 JxfsPINExportRSADESEncipheredPublicKey

The JxfsPINExportRSADESEncipheredPublicKey data class contains data returned on *JxfsOperationCompleteEvent* of the *exportRSADESEncipheredPublicKey()* operation. This class is used to export the public part of a RSA keys.

Summary

Implements: **Extends: JxfsType**

Property	**Type**	**Access**	**Initialized after**
keyValue	byte[ ]	R	
EncipherAlgorithm	JxfsPINRSASignatureAlgo	R	
hashAlgorithm	JxfsPINRSAHashAlgorithm	R	
hashData	byte []	R	
RSASignatureAlgorithm	JxfsPINRSASignatureAlgo	R	
signatureKey	JxfsPINRSAKeyType	R	
signature	byte[]	R	

Method	**Return**	**May use after**
get*Property*	*Property*	
set*Property*	*Property*	
JxfsPINExportRSADESEncipheredPublicKey	(Constructor of the class)	

5.42.1 Properties

keyValue Property (R)

Type ***byte []***
Description Contains the exported RSA public key part.

EncipherAlgorithm Property (R)

Type ***JxfsPINRSASignatureAlgo***
Description Specifies the enciphering algorithm used for the creation of the signature for the exported RSA public key.

hashAlgorithm Property (R)

Type ***JxfsPINRSAHashAlgorithms***
Description Specifies the hash algorithm used for the creation of the signature for the exported RSA public key.

hashData Property (R)

Type ***byte[ ]***
Description The Hash data is used to verify if the key is still valid after it has been stored in the PIN. The PIN runs the stored key through the algorithm described by hashAlgorithm. The result should be identical to the value contained within hashData.

RSASignatureAlgorithm Property (R)

Type ***JxfsPINRSASignatureAlgo***
Description Specifies the algorithm used to generate the signature specified in signature property.

signatureKey Property (R)

Type ***JxfsPINRSAKeyType***

Description	Specifies the private key used to generate the signature returned in the signature property

signature Property (R)

Type	***byte [ ]***
Description	Contains the signature of the RSA public Key exported.

5.42.2 Methods

JxfsPINExportRSADESEncipheredPublicKey Constructor

Syntax ***JxfsPINExportRSADESEncipheredPublicKey (byte[ ] keyValue, JxfsPINRSASignatureAlgo encipherAlgorithm, JxfsPINRSAHashAlgorithm hashAlgorithm, byte[] hashData, JxfsPINRSASignatureAlgo RSASignatureAlgorithm, JxfsPINRSAKeyType signatureKey , byte [ ] signature)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *keyValue* is null or not set correctly *encipherAlgorithm* is null or not set correctly. *hashAlgorithm* is null or not set correctly. *hashData* is null. *signatureAlgo* is null or not set correctly. *signatureAlgorithm* is null or not set correctly. *SignatureKey* is null or not set correctly. *signature* is null or not set correctly.

5.43 JxfsPINGenerateRSAKeyPair

The JxfsPINGenerateRSAKeyPair data class contains data required as input for *generateRSAKeyPair()* operation.

Summary

Implements: **Extends: JxfsType**

Property	Type	Access	Initialized after
key	java.lang.String	R/W	
keyUse	JxfsPINKeyUses	R/W	
modulusLength	int	R/W	
exponentValue	JxfsPINRSAExponent	R/W	

Method	Return	May use after
set*Property*	*void*	
JxfsPINGenerateRSAKeyPair	(Constructor of the class)	

5.43.1 Properties

key Property (R/W)

Type	***java.lang.String***
Description	Name of the key pair o be generated.

keyUse Property (R/W)

Type	***JxfsPINKeyUses***
Description	Type of access for which the key is intended to be used.

modulusLength Property (R/W)

Type	***int***
Description	Specifies the length of the modulus of the generated RSA key Pair

exponentValue Property (R/W)

Type	***JxfsPINRSAExponent***
Description	Specifies the value of the exponent of the generated RSA key pair.

5.43.2 Methods

JxfsPINGenerateRSAKeyPair Constructor

Syntax ***JxfsPINGenerateRSAKeyPair (java.lang.String key, JxfsPINKeyUses keyValue, int modulusLength, jxfsPINRSAExponent exponentValue)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *key* is null or not set correctly. *keyUse* is null or not set correctly. *keyValue* is null or not set correctly *modulusLength* is equal to 0. *exponentValue* null or not set correctly.

5.44 JxfsPINExportId

The JxfsPINExportId data class contains data retrieved by the PIN device and which uniquely identifies the PIN device (e.g.: serial number). This data are as output for exportPINId*()* operation.

Summary

Implements: **Extends: JxfsType**

Property	Type	Access	Initialized after
PINId	byte[ ]	R/W	
hashAlgorithm	JxfsPINRSAHashAlgorithm	R/W	
RSASignatureAlgorithm	JxfsPINRSASignatureAlgo	R./W	
signature	byte[]	R/W	

Method	Return	May use after
Get*Property*	*Property*	
Set*Property*	*void*	
JxfsPINExportId	(Constructor of the class)	

5.44.1 Properties

PINId Property (R/W)

Type	***byte [ ]***
Description	Specifies the item that is unique to the PIN device. This data is vendor dependant item

hashAlgorithm Property (R/W)

Type	***JxfsPINRSAHashAlgorithm***
Description	Specifies the hash algorithm used during the creation of the signature of the security item. to decipher the imported key.

RSASignatureAlgorithm Property (R/W)

Type	***JxfsPINRSASignatureAlgo***
Description	Specifies the algorithm used to generate the signature specified in signature property.

signature Property (R/W)

Type	***byte [ ]***
Description	Contains the signature associated with the PINid. The Signature is used to validate the PINid.

5.44.2 Methods

JxfsPINExportId Constructor

Syntax	***JxfsPINExportId (byte [ ] PINId, JxfsPINRSAHashAlgorithm hashalgorithm, JxfsPINRSASignatureAlgo RSASignatureAlgorithm, byte [ ] signature)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

5.45 JxfsPINExportCertificate

The JxfsPINExportCertificate data class contains data required as output for *exportCertificate()* operation. It specifies the certificate type to export and the certificate itself.

Summary

Implements: **Extends: JxfsType**

Property	Type	Access	Initialized after
certificateType	JxfsPINCertificateType	RW	
certificate	byte[]	R/W	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINExportCertificate	(Constructor of the class)	

5.45.1 Properties

certificateType Property (R/W)

Type ***JxfsPINCertificateType.***
Description Specifies that a primary certificate is to be returned.

certificate Property (W/W)

Type ***byte[].***
Description Contains the certificate that is to be loaded. This data should be in a binary encoded PKCS #7 format containing certificate data represented in DER encoded ASN.1 notation

5.45.2 Methods

JxfsPINExportCertificate Constructor

Syntax ***JxfsPINExportCertificate (JxfsPINCertificateType certificateType, byte [] certificate)***
Description Constructor of the class.
Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

5.46 JxfsPINCertificateType

The JxfsPINCertificateType data class contains the type, primary or secondary, of certificate exported from the encryptor.

Summary

Implements: **Extends: JxfsType**

Property	Type	Access	Initialized after
primary	boolean	R/W	
secondary	boolean	R/W	

Method	Return	May use after
get*Property*	*Property*	
JxfsPINCertificateType	(Constructor of the class)	

5.46.1 Properties

primary Property (R/W)

Type	***boolean.***
Description	Specifies that a primary certificate is to be returned.

secondary Property (R/W)

Type	***boolean.***
Description	Specifies that a secondary certificate is to be returned.

5.46.2 Methods

JxfsPINCertificateType Constructor

Syntax ***JxfsPINCertificateType (boolean primary, boolean secondary)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	Meaning
JXFS_E_PARAMETER_INVALID	Both parameters are set to true or to false.

5.47 JxfsPINCertificateKeyType

The JxfsPINCertificateKeyType data class contains data required as input for *exportCertificate()* operation. It specifies the public key to use

Summary

Implements: **Extends: JxfsType**

Property	**Type**	**Access**	**Initialized after**
encryptionKey	boolean	R/W	
verificationKey	boolean	R/W	

Method	**Return**	**May use after**
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINCertificateKeyTyp e	(Constructor of the class)	

5.47.1 Properties

encryptionKey Property (R/W)

Type ***boolean***
Description Specifies the encryption key is to be returned

verificationKey Property (R/W)

Type ***boolean***
Description Specifies the verification key is to be returned.

5.47.2 Methods

JxfsPINCertificateKeyType Constructor

Syntax ***JxfsPINCertificateKeyType (boolean encryptionKey, boolean verificationKey)***
Description Constructor of the class.
Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVA LID	Both parameters are set to true or to false.

5.48 JxfsPINRSAHashAlgorithms

This class provides properties and methods to query which type of hash algorithms is to be processed

Summary

Implements: **Extends : JxfsType**

Property	Type	Access	Initialized after
Hash_NO	boolean	R/W	
Hash_SHA1	boolean	R/W	

Method	Return	May use after
is*Property*	*Property*	
set*Property*	*void*	
JxfsPINRSAHashAlgorithms	(Constructor of the class)	

5.48.1 Properties

Hash_NO Property (R/W)

Type *boolean*

Description Indicates that no hash algorithm is specified. No hash verification will be applied.

Value	**Meaning**
FALSE	This use is not supported.
TRUE	This use is supported.

Hash_SHA1 Property (R/W)

Type *boolean*

Description Indicates that SHA1 algorithm is supported

Value	**Meaning**
FALSE	This use is not supported.
TRUE	This use is supported.

5.48.2 Methods

JxfsPINRSAHashAlgorithms Constructor

Syntax ***JxfsPINRSAHashAlgorithms (boolean hash_NO, boolean hash_SHA1)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	All the parameters are false.

5.49 JxfsPINRSASignatureAlgo

This class provides properties and methods to query which type of RSA Signature algorithms is to be processed

Summary

Implements:	**Extends :**	**JxfsType**

Property	Type	Access	Initialized after
RSASignature_NO	boolean	R/W	
RSASignature_PKCS1_V1_5	boolean	R/W	
RSASignature_PSS	boolean	R/W	

Method	Return	May use after
is*Property*	*Property*	
set*Property*	*void*	
JxfsPINRSASignatureAlgo	(Constructor of the class)	

5.49.1 Properties

RSASignature_NO Property (R/W)

Type	***boolean***
Description	Indicates that no RSA signature algorithm is specified. No signature verification

Value	**Meaning**
FALSE	This use is not supported.
TRUE	This use is supported.

RSASignature_PKCS1_V1_5 Property (R/W)

Type	***boolean***
Description	Indicates that the RSASSA-PKCS1-V.5 algorithm is used.

Value	**Meaning**
FALSE	This use is not supported.
TRUE	This use is supported.

RSASignature_PSS Property (R/W)

Type	***boolean***
Description	Indicates that the RSASSA-PSS algorithm is used

Value	**Meaning**
FALSE	This use is not supported.
TRUE	This use is supported.

5.49.2 Methods

JxfsPINRSASignatureAlgo Constructor

Syntax	***JxfsPINRSASignatureAlgo (boolean RSASignature_NO, boolean RSASignature_PKCS1_V1_5, boolean RSASignature_PSS);***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	All the parameters are false or more than one signature is set to true

5.50 JxfsPINRSAExponent

This class provides properties and methods to query which exponent value of the RSA key pair to be generated

Summary

Implements: **Extends :** **JxfsType**

Property	Type	Access	Initialized after
PIN_Default	boolean	R/W	
PIN_Exponent_1	boolean	R/W	
PIN_Exponent_4	boolean	R/W	
PIN_Exponent_16	boolean	R/W	

Method	Return	May use after
is*Property*	*Property*	
set*Property*	*void*	
JxfsPINRSAExponent	(Constructor of the class)	

5.50.1 Properties

PIN_Default Property (R/W)

Type	***boolean***	
Description	Indicates that the device will decide of the exponent length	
	Value	**Meaning**
	FALSE	This use is not supported.
	TRUE	This use is supported.

PIN_Exponent_1 Property (R/W)

Type	***boolean***	
Description	Indicates that the exponent length is $2^1 + 1$ (3)	
	Value	**Meaning**
	FALSE	This use is not supported.
	TRUE	This use is supported.

PIN_Exponent_4 Property (R/W)

Type	***boolean***	
Description	Indicates that the exponent length is $2^4 + 1$ (17)	
	Value	**Meaning**
	FALSE	This use is not supported.
	TRUE	This use is supported.

PIN_Exponent_16 Property (R/W)

Type	***boolean***	
Description	Indicates that the exponent length is $2^{16} + 1$ (65537)	
	Value	**Meaning**
	FALSE	This use is not supported.
	TRUE	This use is supported.

5.50.2 Methods

JxfsPINRSAExponent Constructor

Syntax	***JxfsPINRSAExponent (boolean PIN_Default, boolean PIN_Exponent_1, boolan PIN_Exponent_4, boolean PIN_Exponent_16);***
Description	Constructor of the class.

Exceptions

Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	All the parameters are false or more than one parameter is set to true.

5.51 JxfsPINRSAKeyVerificationData

The JxfsPINRSAKeyVerificationData data class contains information about the imported RSA Public key.
This data class is returned on *JxfsOperationCompleteEvent* of the *importRSAPublicKey()* operation.

Summary

Implements: **Extends: JxfsType**

Property	Type	Access	Initialized after
hashAlgorithm	JxfsPINRSAHashAlgorithm	R	
hashData	byte []	R	

Method	Return	May use after
get*Property*	*void*	
JxfsPINRSAKeyVerificationData	(Constructor of the class)	

5.51.1 Properties

hashAlgorithm Property (W)

Type	***JxfsPINRSAHashAlgorithms***
Description	Specifies the hash algorithm used to verify and import the RSA public key.

hashData Property (W)

Type	***byte[ ]***
Description	Contains the Hash data value computed when verifying and importing the key.

5.51.2 Methods

JxfsPINRSAKeyVerificationData Constructor

Syntax ***JxfsPINRSAKeyVerificationData (JxfsPINRSAHashAlgorithm hashAlgorithm, byte [ ] hashData)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Any of the following conditions is met: *hashAlgoithm* is null or not set correctly *hashData* is null.

5.52 JxfsPINRSADESkeyVerificationData

The JxfsPINRSADESkeyVerificationData data class contains information about the imported RSA DES enciphered public key.
This data class is returned on *JxfsOperationCompleteEvent* of the *importRSADESEncipherdPublicKey ()* operation.

Summary

Implements: **Extends: JxfsType**

Property	Type	Access	Initialized after
keyLength	JxfsPINRSADESLength	R	
checkMode	JxfsPINRSADESCheck Mode	R	
checkValue	byte []	R	

Method	Return	May use after
get*Property*	*property*	
JxfsPINRSADESkeyVerificationData	(Constructor of the class)	

5.52.1 Properties

keyLength Property (W)

Type	***JxfsPINRSADESLength***
Description	Specifies the length of the key loaded (simple or double).

checkMode Property (W)

Type	***JxfsPINRSADESCheckMode***
Description	Specifies the mode that was use to create the check value.

checkValue Property (W)

Type	***byte []***
Description	Specifies the verification data that can be used for verification of the loaded key.

5.52.2 Methods

JxfsPINRSADESkeyVerificationData Constructor

Syntax	***JxfsPINRSADESkeyVerificationData (JxfsPINRSADESLength keyLength, JxfsPINRSADESCheckMode checkMode, byte [ ] checkValue)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

5.53 JxfsPINRSADESLength

The JxfsPINRSADESLength data specifies the key length that was loaded

.Summary

Implements: **Extends: JxfsType**

Property	Type	Access	Initialized after
single	boolean	R/W	
double	boolean	R/W	

Method	Return	May use after
is*Property*	*Property*	
JxfsPINRSADESLength	(Constructor of the class)	

5.53.1 Properties

single Property (R/W)

Type	***boolean***
Description	Specifies the length of the key loaded is simple

double Property (R/W)

Type	***boolean***
Description	Specifies the length of the key loaded is double

5.53.2 Methods

JxfsPINRSADESLength Constructor

Syntax ***JxfsPINRSADESLength (boolean single, boolean double)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	Meaning
JXFS_E_PARAMETER_INVALID	Both parameters are set to true or to false.

5.54 JxfsPINRSADESCheckMode

The JxfsPINRSADESCheckMode specifies the mode that was used to create the check value.

.Summary

Implements: **Extends: JxfsType**

Property	Type	Access	Initialized after
noCheck	boolean	R/W	
selfCheck	boolean	R/W	
zeroCheck	boolean	R/W	

Method	Return	May use after
is*Property*	*Property*	
JxfsPINRSADESCheckMode	(Constructor of the class)	

5.54.1 Properties

noCheck Property (R/W)

Type	***boolean***
Description	Specifies the no check value provided

selfCheck Property (R/W)

Type	***boolean***
Description	Specifies that the key check value is created by an encryption of the key with itself

zeroCheck Property (R/W)

Type	***boolean***
Description	Specifies that the key check value is created by an encryption of the key with a zero value

5.54.2 Methods

JxfsPINRSADESCheckMode Constructor

Syntax ***JxfsPINRSADESCheckMode (boolean noCheck, boolean selfCheck, boolean zeroCheck)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	Meaning
JXFS_E_PARAMETER_INVALID	All the parameters are set to false or more than one parameter is set to true..

5.55 JxfsPINRSAKeyType

The JxfsPINRSAKeyType data class specifies the private signature to use

Summary

Implements: **Extends: JxfsType**

Property	**Type**	**Access**	**Initialized after**
signatureIssuer	boolean	R/W	
signatureDevice	boolean	R/W	

Method	**Return**	**May use after**
get*Property*	*Property*	
set*Property*	*void*	
JxfsPINRSAKeyType	(Constructor of the class)	

5.55.1 Properties

signatureIssuer Property (R/W)

Type *boolean*

Description Specifies that the issuer RSA private/public key pair is to be used or has been used to sign the exported key.
These issuer public/private key pairs are installed during manufacture process typically is a secure way.

signatureDevice Property (R/W)

Type *boolean*

Description Specifies that the devices unique private/public key pair is to be used or has been used to sign the exported key.
The public/private key pairs are created by the device with the command generateRSAKeyPair.

5.55.2 Methods

JxfsPINRSAKeyType Constructor

Syntax ***JxfsPINRSAKeyType (boolean signatureIssuer , boolean signatureDevice)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	Both parameters are set to false or to true.:

5.56 JxfsPINRemoteKeyLoadModes

This class provides properties and methods to query which remote key loading modes are supported by a secure PIN device service.

Summary

Implements: **Extends :** **JxfsType**

Property	Type	Access	Initialized after
keyLoadCertificate	boolean	R	
keyLoadSignature	boolean	R	
GenerateRSAkeyPair	boolean	R	
keyCheckRSA	boolean	R	

Method	Return	May use after
is*Property*	*Property*	
JxfsPINRemoteKeyLoadModes	(Constructor of the class)	

5.56.1 Properties

keyLoadCertificate Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports key loading using Three-party authentication through Certificates.

Value	Meaning
FALSE	Key loading mode is not supported.
TRUE	Key loading mode is supported.

keyLoadSignature Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports key loading using Two-party authentication through Signatures.

Value	Meaning
FALSE	Key loading mode is not supported.
TRUE	Key loading mode is supported.

GenerateRSAkeyPair Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports the RSA Key pair generation.

Value	Meaning
FALSE	The device does not support generation of RSA key pairs
TRUE	The device does support generation of key pairs.

keyCheckRSA Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports thumbprint calculation for key loading using Two-party authentication through Signatures.

Value	Meaning
FALSE	SHA1 digest calculation is not supported.

TRUE	SHA1 digest calculation is supported.

5.56.2 Methods

JxfsPINRemoteKeyLoadModes Constructor

Syntax ***JxfsPINRemoteKeyLoadModes (boolean keyLoadCertificate, boolean keyLoadSignature, boolean generateRSAkeyPair, boolean keyCheckRSA)***

Description Constructor of the class.

Exceptions Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	All the parameters are false.

5.57 JxfsPINRSAAlgorithm

This class provides properties and methods to query which RSA algorithm are supported by the secure PIN device service.

Summary

Implements: **Extends :** **JxfsType**

Property	Type	Access	Initialized after
cryptRSA_OAEP	boolean	R	
cryptRSA_PKCS_V1_5	boolean	R	
signatureRSA_OAEP	boolean	R	
signatureRSA_PKCS_V1_5	boolean	R	

Method	Return	May use after
is*Property*	*Property*	
JxfsPINRSAAlgorithm	(Constructor of the class)	

5.57.1 Properties

cryptRSA_OAEP Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports the RSA encryption / decryption using the OAEP scheme (Optimal Asymmetric Eencryption Padding)..

Value	**Meaning**
FALSE	RSA OAEP encryption / decryption are not supported.
TRUE	RSA OAEP encryption / decryption are supported.

cryptRSA_PKCS_V1_5_ Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports the RSA encryption / decryption using the PKCS V1.5 scheme (Public-Key Cryptography Standards)..

Value	**Meaning**
FALSE	RSA PKCS encryption / decryption are not supported.
TRUE	RSA PKCS encryption / decryption are supported.

signatureRSA_OAEP _ Property (R)

Type *boolean*
Initial Value Depends on device
Description Indicates if the device supports the creation of a RSA signature using the OAEP scheme (Optimal Asymmetric Encryption Padding)..

Value	**Meaning**
FALSE	RSA OAEP signature creation is not supported.
TRUE	RSA OAEP signature creation is supported.

signatureRSA_ PKCS_V1_5 _ Property (R)

Type	***boolean***
Initial Value	Depends on device
Description	Indicates if the device supports the creation of a RSA signature using the PKCS V1.5 scheme (Optimal Asymmetric Encryption Padding)..

Value	**Meaning**
FALSE	RSA PKCS signature creation is not supported.
TRUE	RSA PKCS signature creation is supported.

5.57.2 Methods

JxfsPINRSAAlgorithm Constructor

Syntax	***JxfsPINRSAAlgorithm (boolean cryptRSA_OAEP, boolean cryptRSA_PKCS_V1_5, boolean signatureRSA_OAEP, boolean signatureRSA_ PKCS_V1_5)***
Description	Constructor of the class.
Exceptions	Some possible JxfsException *value codes*. See section on JxfsExceptions for other JxfsException value codes.

Value	**Meaning**
JXFS_E_PARAMETER_INVALID	All the parameters are false.

6 Codes

6.1 Error Codes

Value	Meaning
JXFS_E_PIN_READ_FAILURE	Read error.
JXFS_E_PIN_KEYINVALID	At least one of the specified active function keys or FDKeys is invalid.
JXFS_E_PIN_NOACTIVEKEYS	No active function key or FDKey specified.
JXFS_E_PIN_KEYNOTSUPPORTED	At least one of the specified active function keys or FDKeys (*activeFKeys* or *activeFDKeys* properties of *readMode* parameter) is not supported by the device service.
JXFS_E_PIN_MINIMUNLENGTH	The *minLength* property is invalid or greater than the *maxLength* property.
JXFS_E_PIN_NO_PIN	PIN has not been entered or has been cleared.
JXFS_E_PIN NOT_ALLOWED	PIN entered by the user is not allowed.
JXFS_E_PIN_KEY_NOT_FOUND	The specified key was not found.
JXFS_E_PIN_KEY_NO_VALUE	The specified key is not loaded.
JXFS_E_PIN_USE_VIOLATION	The specified use is not supported by this key.
JXFS_E_PIN_ACCESS_DENIED	The encryption module is either not initialized or not ready for any vendor specific reason.
JXFS_E_PIN_NOTSUPPORTEDCAP	The requested function is not supported.
JXFS_E_PIN_FORMAT_NOTSUPPORTED	The specified PIN block format is not supported.
JXFS_E_PIN_LENGTH_ERROR	The length of the start value specified is not supported.
JXFS_E_PIN_CRYPTNOTSUPPORTED	The encryption or decryption method is not supported.
JXFS_E_PIN_DUPLICATE_KEY	A key exists with the specified name and cannot be overwritten.

6.2 Status Codes

Value	Meaning
JXFS_S_PIN_KEY	A new key has been loaded/imported into the device's key table.

6.3 Operation Codes

The following codes identify the operation that generated an OperationCompleteEvent or IntermediateEvent:

Value	Method
JXFS_O_PIN_READPIN	*readData, secureReadPIN*
JXFS_O_PIN_CREATEOFFSET	*createOffset*
JXFS_O_PIN_CREATEPINBLOCK	*createPINBlock*
JXFS_O_PIN_VALIDATEPIN	*validatePIN*
JXFS_O_PIN_CREATEOFFSET_SECURE	*createOffsetSecure*
JXFS_O_PIN_CREATEPINBLOCK_SECURE	*createPINBlockSecure*
JXFS_O_PIN_VALIDATEPIN_SE	*validatePINSecure*

CURE	
JXFS_O_PIN_VALIDATEPINCHIP	*validatePINChip*
JXFS_O_PIN_DECRYPT	*decrypt*
JXFS_O_PIN_ENCRYPT	*encrypt*
JXFS_O_PIN_GENMAC	*generateMAC*
JXFS_O_PIN_IMPORTKEY	*importKey*
JXFS_O_PIN_INITIALIZE	*initialize*

The following codes identify the reason for an IntermediateEvent:

Value	Meaning
JXFS_I_PIN_KEY_PRESSED	A key has been pressed.

6.4 Constants

Value	Meaning
JXFS_PIN_FK_FDK01 to JXFS_PIN_FK_FDK32	Codes of function descriptor keys FDKeys.
JXFS_PIN_FK_0	Function key code.
JXFS_PIN_FK_1	Function key code.
JXFS_PIN_FK_2	Function key code.
JXFS_PIN_FK_3	Function key code.
JXFS_PIN_FK_4	Function key code.
JXFS_PIN_FK_5	Function key code.
JXFS_PIN_FK_6	Function key code.
JXFS_PIN_FK_7	Function key code.
JXFS_PIN_FK_8	Function key code.
JXFS_PIN_FK_9	Function key code.
JXFS_PIN_FK_ENTER	Function key code.
JXFS_PIN_FK_CANCEL	Function key code.
JXFS_PIN_FK_CLEAR	Function key code.
JXFS_PIN_FK_BACKSPACE	Function key code.
JXFS_PIN_FK_HELP	Function key code.
JXFS_PIN_FK_DECPOINT	Function key code.
JXFS_PIN_FK_00	Function key code.
JXFS_PIN_FK_000	Function key code.
JXFS_PIN_FK_NONE	Result of a *secureReadPIN()* operation when key is not a function key.
JXFS_PIN_KP_FUNCTION	Key is a Function key.
JXFS_PIN_KP_FDKEY	Key is a Function descriptor key (FDKey).
JXFS_PIN_INPUT_RAW	Each key pressed during an input operation will generate an intermediate event. These events will contain information about pressed keys.
JXFS_PIN_INPUT_COOKED	No intermediate events per key pressed are generated. Data entered during an input operation is provided in an *OperationCompleteEvent* event.
JXFS_PIN_COMP_AUTO	Input operation terminated because *maxLength* was reached.
JXFS_PIN_COMP_FK	A termination key was pressed.
JXFS_PIN_COMP_FDKEY	A termination FDKey was pressed

Value	Meaning
JXFS_PIN_VAL_DES	DES PIN validation.
JXFS_PIN_VAL_EC	EUROCHEQUE PIN validation.
JXFS_PIN_VAL_VISA	VISA PIN validation.

JXFS_PIN_PRES_CLEAR	Clear text presentation of PIN to chip card device.

PIN block formats:

Value	Meaning
JXFS_PIN_FMT_3624	3624.
JXFS_PIN_FMT_ANSI	ANSI.
JXFS_PIN_FMT_ISO0	ISO0.
JXFS_PIN_FMT_ISO1	ISO1.
JXFS_PIN_FMT_EC12	EC12.
JXFS_PIN_FMT_EC13	EC13.
JXFS_PIN_FMT_EC13RAND	EC13, random padding.
JXFS_PIN_FMT_VISA	VISA.
JXFS_PIN_FMT_DIEBOLD	DIEBOLD.
JXFS_PIN_FMT_DIEBOLDC0	DIEBOLD C0.
JXFS_PIN_FMT_EMV	EMV PIN Format

Encryption/decryption algorithms:

Value	Meaning
JXFS_PIN_CRYPT_MODE_DESECB	Electronic Code Book
JXFS_PIN_CRYPT_MODE_DESCBC	Cipher Block Chaining
JXFS_PIN_CRYPT_MODE_DESMAC	MAC calculation using CBC
JXFS_PIN_CRYPT_MODE_DESCFB	Cipher Feed Back
JXFS_PIN_CRYPT_MODE_RSA	RSA Encryption
JXFS_PIN_CRYPT_MODE_ECMA	ECMA Encryption
JXFS_PIN_CRYPT_MODE_TRIDESECB	Triple DES with Electronic Code Book
JXFS_PIN_CRYPT_MODE_TRIDESCBC	Triple DES with Cipher Block Chaining
JXFS_PIN_CRYPT_MODE_TRIDESCFB	Triple DES with Cipher Feed Back
JXFS_PIN_CRYPT_MODE_TRIDESMAC	Triple DES MAC calculation using CBC

7 Appendix A: ZKA Extensions for the Pin Keypad Device Class Interface

This chapter describes a proposal for an extension of the Pin Keypad device class interface to cover the functionality needed to implement the parts of the ZKA 3.0 specification that are necessary for self-service automates.

For the access of the ZKA functionality the appropriate interfaces, classes and events for handling the ISO messages and the HSM data are defined in addition to the current Pin Keypad device class interface specification.

Important Notes:

- This revision of this specification does not define key management procedures; key management is vendor-specific.
- Key space management is customer-specific, and is therefore handled by vendor-specific mechanisms.
- Only numeric PIN pads are handled in this specification.

Multiple HSMs will be handled by multiple device services. If more than one HSM is implemented in one hardware device, these “logical” HSMs may be presented by one instance of a device service. In this case it is a complex device service that offers its services via different “logical” devices.

This specification supports the Hardware Security Module (HSM), which is necessary for the German ZKA Electronic Purse transactions. Furthermore the HSM stores terminal specific data.
This data will be compared against the message data fields (Sent and Received ISO8583 messages) prior to HSM-MAC generation/verification. HSM-MACs are generated/verified only if the message fields match the data stored.

Keys used for cryptographic HSM functions are stored separate from other keys. This must be considered when importing keys.

This version of PinPad complies to the current ZKA specification 3.0. It supports loading and unloading against card account for both card types (Type 0 and Type 1) of the ZKA electronic purse. It also covers the necessary functionality for ‘Loading against other legal tender’.

Key values are passed to the API as binary hexadecimal values, for example:
0123456789ABCDEF = 0x01 0x23 0x45 0x67 0x89 0xAB 0xCD 0xEF

The implementation for the IJxfsPINIso interface is optional for the device service. If a device service is not intended to be used in germany it does not need to implement any of the functionality described in this document. If the device service offers the functionality to support ZKA 3.0 it has to implement the IJxfsPINIso interface.

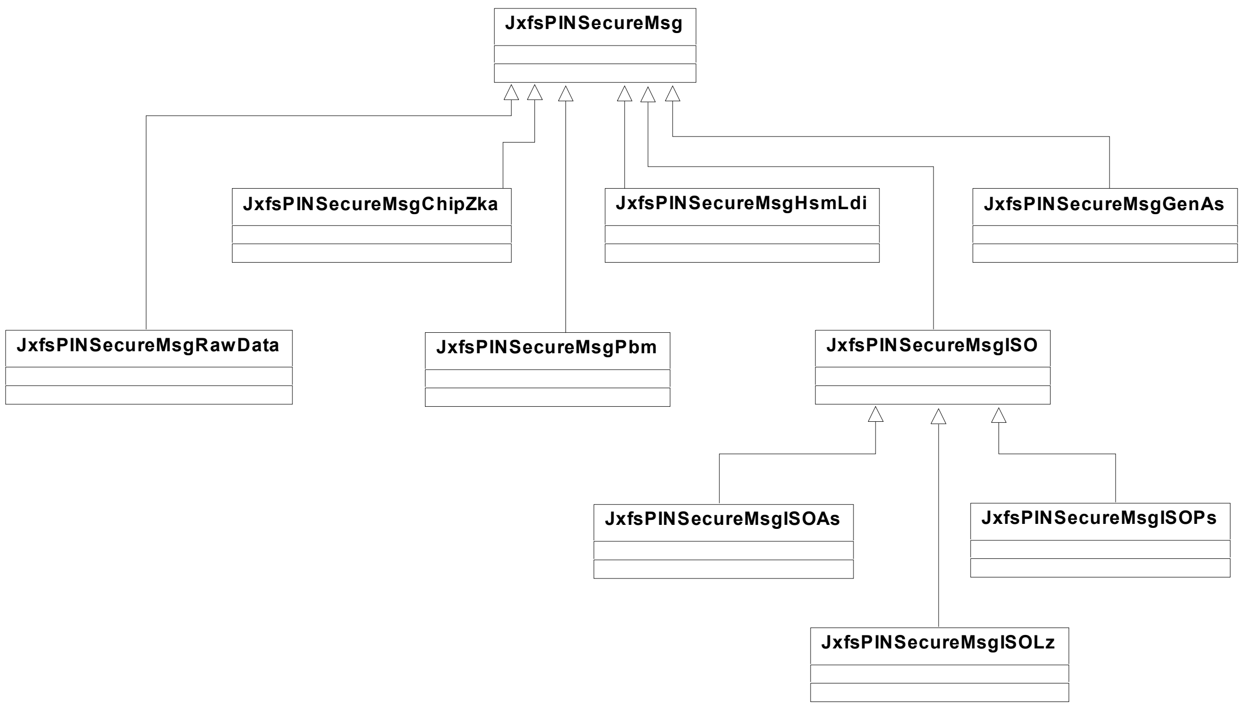
7.1 Class and Interface Summary

The following classes and interfaces are used by the J/XFS Zka extensions:

Class or Inter-face	**Name**	**Description**	**Extends / Implements**
Inter-face	**IJxfsPINIso**	Interface for ISO messages, etc.	--
Class or Inter-face	**Name**	Description	Extends / Implements

Class or Inter-face	Name	Description	Extends / Implements
Inter-face	**IJxfsPINIsoConst**	Interface containing the JXFS constants that are common to the ISO messages interface..	--
Class	**JxfsPINSupportedProtocols**	Capabilities for the IJxfsPINIso interface.	Extends: JxfsType
Class	**JxfsPINSecureMsg**	Representation of a secure raw data message	Extends: JxfsType
Class	**JxfsPINSecureMsgRawData**	Representation of a secure message	Extends: JxfsPINSecureMsg
Class	**JxfsPINSecureMsgChipZka**	Representation of a secure smart card ZKA message	Extends: JxfsPINSecureMsg
Class	**JxfsPINSecureMsgPbm**	Representation of a secure PBM message	Extends: JxfsPINSecureMsg
Class	**JxfsPINSecureMsgHsmLdi**	Representation of a secure LDI message	Extends: JxfsPINSecureMsg
Class	**JxfsPINSecureMsgGenAs**	Representation of a secure Gen AS message	Extends: JxfsPINSecureMsg
Class	**JxfsPINSecureMsgISO**	Representation of aa ISO message	Extends: JxfsPINSecureMsg
Class	**JxfsPINSecureMsgISOPs**	Representation of a secure Ps message	Extends: JxfsPINSecureMsgISO
Class	**JxfsPINSecureMsgISOAs**	Representation of a secure As message	Extends: JxfsPINSecureMsgISO
Class	**JxfsPINSecureMsgISOLz**	Representation of a secure Lz message	Extends: JxfsPINSecureMsgISO
Class	**JxfsPINProtocolSelection**	This class specifies a certain protocol.	Extends: JxfsType
Class	**JxfsPINJournalData**	Object to represent journal data.	Extends: JxfsType
Class	**JxfsPINTData**	Object to represent data to be set in the HSM.	Extends: JxfsType
Class	**JxfsPINIsoSupportedModes**	Object to specify the supported charge modes.	Extends: JxfsType

7.2 Messages



All message classes are derived from the global abstract JxfsPINSecureMsg class. The access to the binary message data is done via the synchronized getMessageData() and setMessageData() methods. If the message data will be modified, the whole message has to be set.

7.3 Classes and Interfaces

7.3.1 IJxfsPINIso

This is the main interface for accessing ZKA specific functionalities.

Summary

Property	Type	Access	Initialized after
supportedProtocols	**JxfsPINSupportedProtocols**	R	successfull open()
supportedJournalingProtocols	**JxfsPINSupportedProtocols**	R	successfull open()
hsmVendor	**String**	R	successfull open()
chargingMode	**JxfsPINIsoSupportedModes**	R	successfull open()

Method	Return	May be used after
get*Property*	*Property*	
secureMsgSend	identificationID	
secureMsgReceive	identificationID	
getJournalData	identificationID	
getHsmTData	identificationID	
setHsmTData	identificationID	
hsmInit	identificationID	

Properties

supportedProtocols (R)

Type	***JxfsPINSupportedProtocols***
Initial Value	*none*
Description	Definition of the supported protocols by the device service (see *JxfsPINSupportedProtocols)*

supportedJournalingProtocols (R)

Type	***JxfsPINSupportedProtocols***
Initial Value	*none*
Description	Definition for which protocols the device service provides journal data (see *JxfsPINSupportedProtocols)*

hsmVendor (R)

Type	***String***
Initial Value	*none*
Description	String identifying the vendor of the HSM module. Examples for this string are "KRONE", "ASCOM", "IBM" or "NCR".

chargingMode (R)

Type	***JxfsPINIsoSupportedModes***
Initial Value	*none*
Description	Specification of the charging modes that are supported by the HSM.

Methods

secureMsgSend

Syntax	***identificationID secureMsgSend(JxfsPINSecureMsg message) throws JxfsException;***
Description	This command handles all messages that should be sent through a

secure messaging to a authorization system, German "Ladezentrale", personalisation system or the chip. The encryption module adds the security relevant fields to the message and returns the modified message in the appropriate OC event. All messages must be presented to the encryptor via this command even if they do not contain security fields in order to keep track of the transaction status in the internal state machine.

Parameter	Type	Name	Meaning
	JxfsPINSecureMsg	message	Specifies the message. The following protocols are supported: JXFS_PIN_PROTISOAS JXFS_PIN_PROTISOLZ JXFS_PIN_PROTISOPS JXFS_PIN_PROTCHIPZKA JXFS_PIN_PROTRAWDATA JXFS_PIN_PROTPBM JXFS_PIN_PROTHSMLDI JXFS_PIN_PROTGENAS

Exceptions No additional exceptions generated.

Events

OperationCompleteEvent
When the operation completes an OperationCompleteEvent will be sent by J/XFS PIN Device Control to all registered OperationCompleteListeners with the following data:

Field	Value
operationID	JXFS_O_PIN_SEND_MSG
identificationID	The corresponding ID
result	JXFS_RC_SUCCESSFUL JXFS_E_PIN_PROTINVALID JXFS_E_PIN_HSMSTATEINVALID JXFS_E_PIN_ACCESSDENIED JXFS_E_PIN_CONTENTINVALID JXFS_E_PIN_FORMATINVALID JXFS_E_PIN_KEYNOTFOUND JXFS_E_PIN_NOPIN
data	JxfsPINSecureMsg object containing the modified message that can now be send to an authorization system, German "Ladezentrale", personalization system or the chip. If the secure message object could not be generated, the data reference is null.

secureMsgReceive

Syntax ***identificationID secureMsgReceive(JxfsPINSecureMsg message) throws JxfsException;***

Description This command handles all messages that are received through a secure messaging from a authorization system, German "Ladezentrale", personalisation system or the chip. The encryption checks the security relevant fields. All messages must be presented to the encryptor via this command even if they do not contain security fields in order to keep track of the transaction status in the internal state machine.

Parameter	Type	Name	Meaning

JxfsPINSecureMsg	message	Specifies the message The following protocols are supported: JXFS_PIN_PROTISOAS JXFS_PIN_PROTISOLZ JXFS_PIN_PROTISOPS JXFS_PIN_PROTCHIPZKA JXFS_PIN_PROTRAWDATA JXFS_PIN_PROTPBM JXFS_PIN_PROTGENAS

Exceptions No additional exceptions generated.

Events

OperationCompleteEvent

When the operation completes an OperationCompleteEvent will be sent by J/XFS PIN Device Control to all registered OperationCompleteListeners with the following data:

Field	**Value**
operationID	JXFS_O_PIN_RECEIVE_MSG
identificationID	The corresponding ID
result	JXFS_RC_SUCCESSFUL JXFS_E_PIN_PROTINVALID JXFS_E_PIN_MACINVALID JXFS_E_PIN_HSMSTATEINVALID JXFS_E_PIN_ACCESSDENIED JXFS_E_PIN_FORMATINVALID JXFS_E_PIN_CONTENTINVALID JXFS_E_PIN_KEYNOTFOUND
data	none.

getJournalData

Syntax ***identificationID getJournalData(JxfsPINProtocolSelection protocol) throws JxfsException;***

Description This command is used to get journal data from the encryptor module. It retrieves cryptographically secured information about the result of the last transaction that was done with the indicated protocol. When the device service supports journaling (see supportedJournalingProtocols) then it is impossible to do any secureMsgSend/secureMsgReceive method calls with this protocol, unless the journal data is retrieved. It is possible – especially after restarting a system – to get the same journal data again.
Calling this method is obligatory between transactions and after failures as it can be used by the device service to initialize its internal state machine.

Parameter

Type	Name	Meaning
JxfsPINProtocolSelection	protocol	Specifies the protocol. Only the ISOAS, ISOLZ, ISOPS or PBM protocols are supported for this method.

Exceptions No additional exceptions generated.

Events

OperationCompleteEvent
When the operation completes an OperationCompleteEvent will be sent by J/XFS PIN Device Control to all registered OperationCompleteListeners with the following data:

Field	Value
operationID	JXFS_O_PIN_GET_JOURNAL
identificationID	The corresponding ID
result	JXFS_RC_SUCCESSFUL JXFS_E_PIN_PROTINVALID JXFS_E_PIN_HSMSTATEINVALID JXFS_E_PIN_ACCESSDENIED
data	JxfsPINJournalData object, if the operation succeeded. null, if the data could not be retrieved.

getHsmTData

Syntax ***identificationID getHsmTData() throws JxfsException;***

Description This function allows to get the current HSM terminal data except keys, trace number and session key index. The data is provided as a series of "tag/length/value" items in the tData class.

Exceptions No additional exceptions generated.

Events

OperationCompleteEvent
When the operation completes an OperationCompleteEvent will be sent by J/XFS PIN Device Control to all registered OperationCompleteListeners with the following data:

Field	Value
operationID	JXFS_O_PIN_HSM_GET_TDATA
identificationID	The corresponding ID
result	JXFS_RC_SUCCESSFUL JXFS_E_PIN_HSMSTATEINVALID JXFS_E_PIN_ACCESSDENIED
data	If the operation was successful the data is the terminal data as an instance of the JxfsPINTData class. This value is null, if the data could not be retrieved.

setHsmTData

Syntax ***identificationID setHsmTData(JxfsPINTData tData) throws JxfsException;***

Description	This function allows to set the HSM terminal data except keys, trace number and session key index. The data must be provided as a series of "tag/length/value" items in the Data class.

Parameter

Type	**Name**	**Meaning**
JxfsPINTData	tData	Specifies the values to set

Exceptions No additional exceptions generated.

Events

OperationCompleteEvent
When the operation completes an OperationCompleteEvent will be sent by J/XFS PIN Device Control to all registered OperationCompleteListeners with the following data:

Field	**Value**
operationID	JXFS_O_PIN_HSM_SET_TDATA
identificationID	The corresponding ID
result	JXFS_RC_SUCCESSFUL JXFS_E_PIN_HSMSTATEINVALID JXFS_E_PIN_ACCESSDENIED
data	none

hsmInit

Syntax ***identificationID hsmInit(JxfsPINHsmInitData hsmInitData) throws JxfsException;***

Description This command is used to set an HSM out of order. At the same time the online time can be set to control when the online dialog will be started to initialize the HSM again.

Parameter

Type	**Name**	**Meaning**
JxfsPINHsmInitData	hsmInitData	Specifies the data for the initialization.

Exceptions No additional exceptions generated.

Events

OperationCompleteEvent
When the operation completes an OperationCompleteEvent will be sent by J/XFS PIN Device Control to all registered OperationCompleteListeners with the following data:

Field	**Value**
operationID	JXFS_O_PIN_HSM_INIT
identificationID	The corresponding ID
result	JXFS_RC_SUCCESSFUL JXFS_E_PIN_MODENOTSUPPORTED JXFS_E_PIN_HSMSTATEINVALID
data	none.

7.3.2 JxfsPINSecureMsg

This class defines a secure message. As every specific message has its own class type, this class is abstract.

Summary

Implements : *Serializable, Clonable* **Extends :** *JxfsType*

Property	Type	Access	Initialized after
messageData	byte[]		

Constructor	Parameter	Parameter-Type
JxfsPINSecureMsg	messageData	byte[]

Method	Return	May be used after
get*Property*	*Property*	
set*Property*		

Event	May occur after

none	

Properties

messageData (R)

Type	***byte[]***
Initial Value	none
Description	Message data. null is permitted in the special case that during the message receive no response was received from the communication partner during a specified time period. This exception is necessary to set the internal state machine to the correct state.

7.3.3 JxfsPINSecureMsgRawData

This class defines a secure message with raw data contents that may be used by a vendor for specific purpose.

Summary

Implements : *Serializable, Clonable* **Extends :** *JxfsPINSecureMsg*

Property	Type	Access	Initialized after
none	none		

Constructor	Parameter	Parameter-Type
JxfsPINSecureMsgRawData	messageData	byte[]

Method	Return	May be used after
get*Property*	*Property*	
set*Property*		

Event	May occur after
none	

7.3.4 JxfsPINSecureMsgChipZka

This class defines a secure message for chip card data.

Summary

Implements : *Serializable, Clonable* **Extends :** *JxfsPINSecureMsg*

Property	Type	Access	Initialized after
none	none		

Constructor	Parameter	Parameter-Type
JxfsPINSecureMsgChipZka	messageData	byte[]

Method	Return	May be used after
get*Property*	*Property*	
set*Property*		

Event	May occur after
none	

7.3.5 JxfsPINSecureMsgPbm

This class defines a secure message for embedded PBM protocol data.

Summary

Implements : *Serializable, Clonable* **Extends :** *JxfsPINSecureMsg*

Property	Type	Access	Initialized after
none	none		

Constructor	Parameter	Parameter-Type
JxfsPINSecureMsgPbm	messageData	byte[]

Method	Return	May be used after
get*Property*	*Property*	
set*Property*		

Event	May occur after
none	

7.3.6 JxfsPINSecureMsgHsmLdi

This class defines a secure message that contains LDI Information.

Summary

Implements : *Serializable, Clonable* **Extends :** *JxfsPINSecureMsg*

Property	Type	Access	Initialized after
none	none		

Constructor	Parameter	Parameter-Type
JxfsPINSecureMsgHsmLdi	messageData	byte[]

Method	Return	May be used after
get*Property*	*Property*	
set*Property*		

Event	May occur after
none	

7.3.7 JxfsPINSecureMsgGenAs

This class defines a secure message that contains PAC/MAC information for non-ISO8583 message formats.

Summary

Implements : *Serializable, Clonable* **Extends :** *JxfsPINSecureMsg*

Property	Type	Access	Initialized after
none	none		

Constructor	Parameter	Parameter-Type
JxfsPINSecureMsgGenAs	messageData	byte[]

Method	Return	May be used after
get*Property*	*Property*	
set*Property*		

Event	May occur after
none	

7.3.8 JxfsPINSecureMsgISO

This abstract class defines the base for all ISO 8583 secure messages.

Summary

Implements : *Serializable, Clonable* **Extends :** *JxfsPINSecureMsg*

Property	Type	Access	Initialized after
none	none		

Constructor	Parameter	Parameter-Type
none	none	none

Method	Return	May be used after
get*Property*	*Property*	
set*Property*		

Event	May occur after
none	

7.3.9 JxfsPINSecureMsgISOAs

This class defines a secure message that contains an ISO 8583 secure message for the authorization system.

Summary

Implements : *Serializable, Clonable* **Extends :** *JxfsPINSecureMsgISO*

Property	Type	Access	Initialized after
none	none		

Constructor	Parameter	Parameter-Type
JxfsPINSecureMsgISOAs	messageData	byte[]

Method	Return	May be used after
get*Property*	*Property*	
set*Property*		

Event	May occur after
none	

7.3.10 JxfsPINSecureMsgISOLz

This class defines a secure message that contains an ISO 8583 secure message for the german "Ladezentrale".

Summary

Implements : *Serializable, Clonable* **Extends :** *JxfsPINSecureMsgISO*

Property	Type	Access	Initialized after
none	none		

Constructor	Parameter	Parameter-Type
JxfsPINSecureMsgISOLz	messageData	byte[]

Method	Return	May be used after
get*Property*	*Property*	
set*Property*		

Event	May occur after
none	

7.3.11 JxfsPINSecureMsgISOPs

This class defines a secure message that contains an ISO 8583 secure message for the personalization system

Summary

Implements : *Serializable, Clonable* **Extends :** *JxfsPINSecureMsgISO*

Property	Type	Access	Initialized after
none	none		

Constructor	Parameter	Parameter-Type
JxfsPINSecureMsgISOPs	messageData	byte[]

Method	Return	May be used after
get*Property*	*Property*	
set*Property*		

Event	May occur after
none	

7.3.12 JxfsPINSupportedProtocols

This class is used to specify the capabilites (supported protocol types).

Summary

Implements : *Serializable* **Extends :** *JxfsType*

Property	Type	Access	Initialized after
protocolIsoAs	boolean	R	
protocolIsoLz	boolean	R	
protocolIsoPs	boolean	R	
protocolChipZka	boolean	R	
protocolRawData	boolean	R	
protocolPbm	boolean	R	
protocolHsmLdi	boolean	R	
protocolGenAs	boolean	R	

Constructors	Parameter	Parameter-Type
JxfsPINSupportedProtocols	protocolIsoAs	boolean
	protocolIsoLz	boolean
	protocolIsoPs	boolean
	protocolChipZka	boolean
	protocolRawData	boolean
	protocolPbm	boolean
	protocolHsmLdi	boolean
JxfsPINSupportedProtocols	protocolIsoAs	boolean
	protocolIsoLz	boolean
	protocolIsoPs	boolean
	protocolChipZka	boolean
	protocolRawData	boolean
	protocolPbm	boolean
	protocolHsmLdi	boolean
	protocolGenAs	boolean

Method	Return	May be used after
isProtocolIsoAs	boolean	
isProtocolIsoLz	boolean	
isProtocolIsoPs	boolean	
isProtocolChipZka	boolean	
isProtocolRawData	boolean	
isProtocolPbm	boolean	
isProtocolHsmLdi	boolean	
isProtocolGenAs	boolean	

Event	May occur after
none	

Properties

protocolIsoAs (R)

Type	***boolean***
Initial Value	none
Description	Specifies if the ISO 8583 protocol functionality for the authorization system is supported.

protocolIsoLz (R)

Type	***boolean***
Initial Value	none
Description	Specifies if the ISO 8583 protocol functionality for the german "Ladezentrale" is supported.

protocolIsoPs (R)

Type	***boolean***
Initial Value	none
Description	Specifies if the ISO 8583 protocol functionality for the personalisation system is supported.

protocolChipZka (R)

Type	***boolean***
Initial Value	none
Description	Specifies if the ZKA chipcard protocol functionality is supported

protocolRawData (R)

Type	***boolean***
Initial Value	none
Description	Specifies if the raw data protocol functionality is supported.

protocolPbm (R)

Type	***boolean***
Initial Value	none
Description	Specifies if the PBM protocol functionality is supported.

protocolHsmLdi (R)

Type	***boolean***
Initial Value	none
Description	Specifies if the Hsm LDI protocol functionality is supported.

protocolGenAs (R)

Type	***boolean***
Initial Value	none
Description	Specifies if the Gen AS protocol functionality is supported.

Methods

isProtocolIsoAs

Syntax	***boolean isProtocolAs();***
Description	This method returns true, if the protocolIsoAs property is set to true.

isProtocolIsoLz

Syntax	***boolean isProtocolLz();***
Description	This method returns true, if the protocolIsoLz property is set to true.

isProtocolIsoPs

Syntax	***boolean isProtocolPs();***
Description	This method returns true, if the protocolIsoPs property is set to true.

isProtocolChipZka

Syntax	***boolean isProtocolChipZka();***
Description	This method returns true, if the protocolChipZka property is set to true.

isProtocolRawData

Syntax	***boolean isProtocolRawData();***
Description	This method returns true, if the protocolRawData property is set to true.

isProtocolPbm

Syntax	***boolean isProtocolPbm();***
Description	This method returns true, if the protocolPbm property is set to true.

isProtocolHsmLdi

Syntax	***boolean isProtocolHsmLdi();***
Description	This method returns true, if the protocolHsmLdi property is set to true.

isProtocolGenAs

Syntax	***boolean isProtocolGenAs();***
Description	This method returns true, if the protocolGenAs property is set to true.

7.3.13 JxfsPINProtocolSelection

This class is used to select a certain protocol.

Summary

Implements : *Serializable* **Extends :** *JxfsType*

Property	Type	Access	Initialized after
protocol	int	R	

Constructor	Parameter	Parameter-Type
JxfsPINProtocolSelection	protocol	int

Method	Return	May be used after
get*Property*	*Property*	

Event	May occur after
none	

Properties

protocol (R)

Type	***int***
Initial Value	none
Description	Specifies the selected protocol to be used.

Value	**Meaning**
JXFS_PIN_PROTISOAS	ISO 8583 protocol for the authorization system.
JXFS_PIN_PROTISOLZ	ISO 8583 protocol for the german "Ladezentrale".
JXFS_PIN_PROTISOPS	ISO 8583 protocol for the personalization system
JXFS_PIN_PROTCHIPZKA	ZKA chip protocol
JXFS_PIN_PROTRAWDATA	Raw data protocol
JXFS_PIN_PROTPBM	PBM protocol
JXFS_PIN_PROTHSMLDI	HSM LDI protocol
JXFS_PIN_PROTGENAS	Gen AS protocol

Constructor

JxfsPINProtocolSelection

Syntax ***JxfsPINProtocolSelection(int protocol) throws JxfsException;***

Description .

Parameter	**Type**	**Name**	**Meaning**
	int	protocol	Specifies the protocol to be used

Exceptions

JXFS_E_PIN_PROTINVALID	Unknown value for the protocol.

7.3.14 JxfsPINTData

This class defines tag/length/value items with no separator to be set/get in the HSM. The methods to access the data are synchronized.

Summary

Implements : *Serializable* **Extends :** *JxfsType*

Property	Type	Access	Initialized after
data	byte[]	R	

Constructor	Parameter	Parameter-Type
JxfsPINTData	data	byte[]

Method	Return	May be used after
get*Property*	*Property*	
getTag	*byte[]*	
setTag	*Property*	

Event	May occur after
none	

Properties

data (R)

Type	***byte[]***
Initial Value	None

Description

Specifies a set of tag/length/value items where each item consists of

- one byte tag (see list of tags below)
- one byte specifiying the length of the following data as an unsigned binary number
- n bytes of data

tag (hexdec)	Format	Length	Meaning
C2	BCD	4	Terminal ID ISO BMP 41
C3	BCD	4	Blank Code ISO BMP 42 (rightmost 4 bytes)
C4	BCD	9	Account data for terminal account ISO BMP 60 (loading against other card)
C5	BCD	9	Account data for fee account ISO BMP 60 ("Laden vom Kartenkonto")
C6	EBCDIC	40	Terminal Location ISO BMP 43
C7	ASCII	3	Terminal Currency
C8	BCD	7	Online date and time (YYYYMMDDHHMMSS) ISO_BMP 61
C9	BCD	4	Minimum load fee in units of 1/100 of terminal currency, checked against leftmost 4 bytes of ISO BMP 42
CA	BCD	4	Maximum load fee in units of 1/100 of terminal currency, checked against leftmost 4 bytes of ISO BMP 42
CB	BIN	3	Logical HSM binary coded serial number (starts with 1; 0 means that there are no logical HSMs)
CC	EBCDIC	16	ZKA ID (is filled during the preinitialisation of the HSM)
CD	BIN	1	HSM status (1 = irreversibly out of order 2 = out of order, K_UR is not loaded 3 = not pre-initialized, K_UR is loaded 4 = pre-initialized, K_INIT is loaded 5 = initialized/personalized, K_PERS is loaded)

Constructor

JxfsPINTData

Syntax ***JxfsPINTData(byte data[]) throws JxfsException;***

Description If the data is either null or is not in a valid format, the JXFS_E_PIN_INVALID_TAG exception is thrown.

Parameter

Type	Name	Meaning
byte[]	data	Specifies the Tag data.

Methods

getTag

Syntax	***synchronized byte[] getTag(byte tag) throws JxfsException;***		
Description	This method returns the contents of the specified tag.		
Parameter	**Type**	**Name**	**Meaning**
	byte	tag	Specifies the tag.
Exceptions			
	Value		**Meaning**
	JXFS_E_PIN_INVALID_TAG		The specified tag does not exist in the data.

setTag

Syntax	***synchronized void setTag(byte tag, byte value[]) throws JxfsException;***		
Parameter	**Type**	**Name**	**Meaning**
	byte	tag	Specifies the tag.
	byte[]	value	Specifies the value of the tag to be set.
Description	This method sets the appropriate tag in the message. Any same tag will be overwritten.		
Exceptions			
	Value		**Meaning**
	JXFS_E_PIN_INVALID_TAG		The specified tag is invalid like in the case of zero or more than 255 bytes of data.

7.3.15 JxfsPINJournalData

This class defines journal data from the HSM.

Summary

Implements : *Serializable* **Extends :** *JxfsType*

Property	Type	Access	Initialized after
data	byte[]	R	

Constructor	Parameter	Parameter-Type
JxfsPINJournalData	data	byte[]

Method	Return	May be used after
get*Property*	*Property*	

Event	May occur after
none	

Properties

data (R)

Type	***byte[]***
Initial Value	none
Description	Journal Data.

Constructor

JxfsPINJournalData

Syntax	***JxfsPINJournalData(byte data[]) throws JxfsException;***

Description	If the data is null, the JXFS_E_PARAMETER_INVALID exception is thrown.		
Parameter	**Type**	**Name**	**Meaning**
	byte[]	data	Specifies the journal data.

7.3.16 JxfsPINIsoSupportedModes

This class is used to specify the supported charging modes.

Summary

Implements : *Serializable* **Extends :** *JxfsType*

Property	Type	Access	Initialized after
chargeAccount	boolean	R	
chargeCreditCard	boolean	R	
chargeECcard	boolean	R	
chargeCash	boolean	R	
chargeInternationalECcard	boolean	R	
dischargeECcard	boolean	R	

Constructor	Parameter	Parameter-Type
JxfsPINIsoSupportedModes	chargeAccount	boolean
	chargeCreditCard	boolean
	chargeECcard	boolean
	chargeCash	boolean
	chargeInternationalECcard	boolean
	dischargeECcard	boolean

Method	Return	May be used after
isChargeAccount	boolean	
isChargeCreditCard	boolean	
isChargeECcard	boolean	
isChargeCash	boolean	
isChargeInternationalECcard	boolean	
isDischargeECcard	boolean	

Event	May occur after
none	

Properties

chargeAccount (R)

Type	***boolean***
Initial Value	none
Description	Specifies if charging against an account is supported.

chargeCreditCard (R)

Type	***boolean***
Initial Value	none
Description	Specifies if charging against a credit card is supported.

chargeECcard (R)

Type	***boolean***
Initial Value	none
Description	Specifies if charging against an EC-card is supported.

chargeCash (R)

Type	***boolean***
Initial Value	none

Description	Specifies if charging against cash is supported.

chargeInternationalECcard (R)

Type	***boolean***
Initial Value	none
Description	Specifies if charging against an international EC-card is supported.

dischargeECcard (R)

Type	***boolean***
Initial Value	none
Description	Specifies if discharging against an account of a EC-card is supported.

Methods

isChargeAccount

Syntax	***boolean isChargeAccount();***
Description	This method returns true, if the chargeAccount property is set to true.

isChargeCreditCard

Syntax	***boolean isChargeCreditCard();***
Description	This method returns true, if the chargeCreditCard property is set to true.

isChargeECcard

Syntax	***boolean isChargeECcard();***
Description	This method returns true, if the chargeECcard property is set to true.

isChargeCash

Syntax	***boolean isChargeCash();***
Description	This method returns true, if the chargeCash property is set to true.

isChargeInternationalECcard

Syntax	***boolean isChargeInternationalECcard();***
Description	This method returns true, if the chargeInternationalECcard property is set to true.

isDischargeECcard

Syntax	***boolean isDishargeECcard();***
Description	This method returns true, if the dischargeECcard property is set to true.

7.3.17 JxfsPINHsmInitData

This class defines the necessary data for setting an HSM out of order.

Summary

Implements : *Serializable* **Extends :** *JxfsType*

Property	Type	Access	Initialized after
initMode	int	R	
onlineTime	byte[]	R	

Constructor	Parameter	Parameter-Type
JxfsPINHsmInitData	initMode	int
	onlineTime	byte[]

Method	Return	May be used after
get*Property*	*Property*	

Event	May occur after
none	

Properties

initMode (R)

Type	*int*
Initial Value	none
Description	Specifies the initialization mode as one of the following values:

Value	**Meaning**
JXFS_PIN_INITTEMP	Initialize the HSM temporarily (K_UR remains loaded)
JXFS_PIN_INITDEFINITE	Initialize the HSM definitely (K_UR is deleted)
JXFS_PIN_INITIRREVERSIBLE	Initialize the HSM irreversibly (can only be restored by the vendor)

onlineTime (R)

Type	***byte[]***
Initial Value	none
Description	Specifies the Online date and time in the format YYYYMMDDHHMMSS like in ISO BMP 61 as BCD packed characters. This parameter is ignored when the init mode equals JXFS_PIN_INITDEFINITE or JXFS_PIN_INITIRREVERSIBLE. If this parameter is null, the length of the array is zero or the value is 0x00 0x00 0x00 0x00 0x00 0x00 0x00 the online time will be set to a value in the past.

7.4 Codes

7.4.1 Error Codes

Value	Meaning
JXFS_E_PIN_PROTINVALID	The specified protocol is invalid
JXFS_E_PIN_HSMSTATEINVALID	The HSM is not in a correct state to handle this message
JXFS_E_PIN_MACINVALID	The MAC of the message is not correct
JXFS_E_PIN_ACCESSDENIED	The encryption module is either not initialized or not ready for any vendor specific reason.
JXFS_E_PIN_FORMATINVALID	The format of the message is invalid.
JXFS_E_PIN_CONTENTINVALID	The contents of one of the security relevant fields are invalid.
JXFS_E_PIN_INVALID_TAG	The value of the tag data is invalid.
JXFS_E_PIN_KEYNOTFOUND	No key was found for PAC/MAC generation.
JXFS_E_PIN_NOPIN	No PIN or insufficient PIN-digits have been entered.

7.5 Status Events

Value	Meaning
JXFS_S_PIN_OPT_REQUIRED	This status event indicates that the online data/time stored in a HSM has been reached. As there are no more details available, the

	details property of this status event is null. This event may be triggered by the clock reaching a previously stored online time or by the online time being set to a time that lies in the past. The online time may be set by the setHsmTData method or by a secureMsgReceive method that contains a message from a host system containing a new online date/time. The event does not mean that any keys or other data in the HSM is out of date now. It just indicates that the terminal should communicate with a "Personalisierungsstelle" as soon as possible using the methods secureMsgSend / secureMsgReceive and the ISOPS protocol.
JXFS_S_PIN_HSM_TDATA_CHANGED	This event indicates that one of the values of the terminal data has changed (these are the data that can be set using setHsmTData). I.e. this event will be sent especially when the online time or the HSM status is changed because of a hsmInit command or an OPT online dialog (secureMsgSend / secureMsgReceive with JXFS_PIN_PROTPS). The data is a JxfsPINTData object.

7.6 German ZKA GeldKarte

7.6.1 Source of ZKA information

The PIN device is able to handle the German "GeldKarte", which is an electronic purse specified by the ZKA (Zentraler Kreditausschuß).

For anyone attempting to write an application that handles these chipcards, it is essential to read and understand the specifications published by

Bank-Verlag, Köln
Postfach 30 01 91
D-50771 Köln
Phone: +49 221 5490-0
Fax: +49 221 5490-120

7.6.2 How to use the secureMsg methods

This is to describe how an application should use the secureMsgSend and secureMessageReceive commands for transactions involving chipcards with a German ZKA GeldKarte chip.

- Applications must call secureMsgSend for every command they send to the chip or to a host system, including those commands that do not actually require secure messaging. This enables the device service to remember security-relevant data that may be needed or checked later in the transaction.
- Applications must pass a complete message as input to secureMsgSend, with all fields - including those that will be filled by the device service - being present in the correct length. All fields that are not filled by the device service must be filled with the ultimate values in order to enable MACing by the device service.
- Every command secureMsgSend that an application issues must be followed by exactly one command secureMessageReceive that informs the device service about the response from the chip or host. If no response is received (timeout or communication failure) the application must issue a

secureMessageReceive command with no data (message == null) to inform the device service about this fact.

- If a system is restarted after a secureMsgSend was issued to the device service but before the secureMessageReceive was issued, the restart has the same effect as a secureMessageReceive command with message ==null.
- Between a secureMsgSend and the corresponding secureMessageReceive no secureMsgSend with the same protocol value must be issued. Other executional commands of the PIN device – including secureMsgSend / Receive with different protocol – may be used.

7.6.3 Protocol JXFS_PIN_PROTISOAS

This protocol handles ISO8583 messages between an ATM and an authorization system (AS).

Only messages in the new ISO format, with new PAC/MAC-format using session keys and Triple-DES are supported

Authorization messages may be used to dispense the amount authorized in cash or to load the amount into an electronic purse (GeldKarte).

For loading a GeldKarte the only type of authorization supported is a transaction originating from track 3 of a German ec-card (message types 0200/0210 for authorization and 0400/0410 for reversal)

For dispensing cash, transactions originating from international cards (message types 0100/0110 and 0400/0410) are supported as well.
The following bitmap positions are filled by the device service:

- BMP11 Trace-Nummer
- BMP52 PAC
- BMP57 Verschlüsselungsparameter (only the challenge values RND_{MES} and RND_{PAC})
- BMP64 MAC

These bitmaps have to be present and the corresponding flag has to be set in the primary bitmap when the ISO message is passed to the HSM.

The following bitmap positions are checked by the device service and have to be filled by the application:

- Nachrichtentyp
- BMP3 Abwicklungskennzeichen (only for GeldKarte, not for cash)
- BMP4 Transaktionsbetrag (only for GeldKarte, not for cash)
- BMP41 Terminal-ID
- BMP42 Betreiber-BLZ

For a documentation of authorization messages see:

Regelwerk für das deutsche ec-Geldautomaten-System
Stand: 22. Nov. 1999

Bank-Verlag, Köln
Autorisierungszentrale GA/POS der privaten Banken
Spezifikation für GA-Betreiber
Version 3.12
31. Mai 2000

dvg Hannover
Schnittstellenbeschreibung für Autorisierungsanfragen bei nationalen GA-Verfügungen unter Verwendung der Spur 3
Version 2.5
Stand: 15.03.2000

dvg Hannover
Schnittstellenbeschreibung für Autorisierungsanfragen bei internationalen Verfügungen unter Verwendung der Spur 2
Version 2.6
Stand: 30.03.2000

7.6.4 Protocol JXFS_PIN_PROTISOLZ

This protocol handles ISO8583 messages between a „Ladeterminal" and a „Ladezentrale" (LZ).

Only messages in the new ISO format, with new MAC-format using session keys and Triple-DES are supported.

Both types of GeldKarte chip (type 0 = DEM, type 1 = EUR) are supported.

The following bitmap positions are filled by the device service:

- BMP11: Trace-Nummer
- BMP57: Verschlüsselungsparameter (only the challenge value RND_{MES})
- BMP64: MAC

These bitmaps have to be present and the corresponding flag has to be set in the primary bitmap when the ISO message is passed to the HSM.

The following bitmap positions are checked by the device service and have to be filled by the application:

- Nachrichtentyp
- BMP3: Abwicklungskennzeichen
- BMP4: Transaktionsbetrag
- BMP12: Uhrzeit
- BMP13: Datum
- BMP25: Konditionscode
- BMP41: Terminal-ID
- BMP42: Betreiber-BLZ (caution: "Ladeentgelt" also in BMP42 is not set by the EPP)
- BMP61: Online-Zeitpunkt
- BMP62: Chipdaten

The following bitmap positions are only checked if they are available:

- BMP43: Standort
- BMP60: Kontodaten Ladeterminal

For a documentation of the Ladezentrale interface see:
ZKA / Bank-Verlag, Köln
Schnittstellenspezifikation für die ec-Karte mit Chip
Geldkarte Ladeterminals
Version 3.0
2. 4. 1998

7.6.5 Protocol JXFS_PIN_PROTISOPS

This protocol handles ISO8583 messages between a terminal and a "Personalisierungsstelle" (PS). These messages are about OPT.

The device service creates the whole message with secureMsgSend, including message type and bitmap.

For a documentation of the Personalisierungsstelle interface see:
ZKA / Bank-Verlag, Köln
Schnittstellenspezifikation für die ec-Karte mit Chip
Online-Personalisierung von Terminal-HSMs
Version 3.0
2. 4. 1998

7.6.6 Protocol JXFS_PIN_PROTCHIPZKA

This protocol is intended to handle messages between the application and a GeldKarte.

Both types of GeldKarte are supported.

Both types of load transactions ("Laden vom Kartenkonto" and "Laden gegen andere Zahlungsmittel") are supported.

See the chapter "Command Sequence" below for the actions that device service s take for the various chip card commands.

Only the command APDUs to and the response APDUs from the chip must be passed to the device service, the ATR (answer to reset) data from the chip is not passed to the device service.

For a documentation of the chip commands used to load a GeldKarte see:
ZKA / Bank-Verlag, Köln
Schnittstellenspezifikation für die ec-Karte mit Chip
Ladeterminals
Version 3.0
2. 4. 1998

ZKA / Bank-Verlag, Köln
Schnittstellenspezifikation für die ZKA-Chipkarte
Online-Vor-Initialisierung und Online-Anzeige
einer Außerbetriebnahme von Terminal-HSMs
Version 1.0
04.08.2000

7.6.7 Protocol JXFS_PIN_PROTRAWDATA

This protocol is intended for vendor-specific purposes. Generally the use of this protocol is not recommended and should be restricted to issues that are impossible to handle otherwise.

For example a HSM that requires vendor-specific, cryptographically secured data formats for importing keys or terminal data may use this protocol.

Applicaton programmers should be aware that the use of this command may prevent their applications from running on different hardware.

7.6.8 Protocol JXFS_PIN_PROTPBM

This protocol handles host messages between a terminal and a host system, as specified by IBM’s PBM protocol.
For a documentation of this protocol see:
IBM 473x / Personal Banking Machines / Programmer’s Reference
Volume 1 - 4 / GA19-5510 - GA19-5513

Some additions are defined to the PBM protocol in order to satisfy the German ZKA 3.0 PAC/MAC standard. See:
Diebold‘s and IBM‘s Specification for support of Online Preinitialization and Personalization of Terminal HSMs (OPT) and support for the PAC/MAC standards for th 473x Protocol.
Diebold USA, Revision 1.8, revised on Jan-03-2001

The commands secureMsgSend and secureMessageReceive handle the PAC and MAC in the VARDATA ‘K’ subfield of transactions records and responses. The MAC in the traditional MACODE field is not affected.
In order to enable the service provider to understand the messages, the application must provide the messages according to the following rules:

- All alphanumeric fields must be coded in EBCDIC
- Pre-Edit (padding and blank compression) must not be done by the application. The service provider will check the MACMODE field and do what has to be done.
- In order to enable the service provider to find the vardata subfield ‘K’, it must be included in the message by the application, with the indicator ‘K’ and its length set.
- Because CARDDATA (track 2) and T3DATA (track 3) fields always take part in the MAC computation for a transaction record, these fields must be included in the message, even if they already

have been sent to the host in a previous transaction record and the CI-Option SHORTREC prevents them from being sent again.

7.6.9 Protocol JXFS_PIN_PROTHSMLDI

With this protocol an application can request information about the personalized OPT groups.

The information returned consists of personalisation record like in BMP62 of an OPT response but without MAC.

Data format:

XX XX VV	group ID and versions number
XX	number of LDIs within the group (binary coded)
...	
first LDI of the group	
...	
last LDI of the group	
XX XX VV	group ID and versions number
...	
etc. for several groups	

Each LDI consists of

NN	Number of the LDI
00	Alg. code
LL	Length of the following data
XX...XX	data of the LDI

The device service must at least return the standard LDI, but can return more LDIs.

7.6.10 Protocol JXFS_PIN_PROTGENAS

This protocol allows one to create a PAC (encrypted Pin-Block) and to create and verify a MAC for a proprietary message. As the device service doesn't know the message format, it cannot complete the message by adding security relevant fields like random values, PAC and MAC, like it does for the protocol JXFS_PIN_PROTISOAS. Only the application is able to place these fields into the proper locations. Using this protocol, an application can generate the PAC and the random values in separate steps, add them to the proprietary send-message, and finally let the device service generate the MAC. The generated MAC can then be added to the send-message as well.
For a received message, the application extracts the MAC and the associated random value and passes them along with the entire message data to the device service for MAC verification.
PAC generation supports Pin-Block ISO-Format 0 and 1.

Command description:
The first byte of the field *messageData* of the JxfsPINSecureMsgPacMac object contains a subcommand, which is used to qualify the type of operation. The remaining bytes of the message data are dependent on the value of the subcommand.

The following sub-commands are defined:

- GeneratePAC (Code 0x01)
  Returns the encrypted Pin-Block together with generation and version values of the Master Key and the PAC random value.

- GetMACRandom (Code 0x02)
  Returns the generation and version values of the Master Key and the MAC random value.

- GenerateMAC (Code 0x03)
  Returns the generated MAC for the message data passed in. Note, that the MAC is generated for exactly the data that is presented (contents and sequence). Data that should not go into the MAC calculation must not be passed in.

- VerifyMAC (Code 0x04)
  Generates a MAC for the data passed in and compares it with the provided MAC value. MAC random value, key generation and key version must be passed in separately.

PROTGENAS Error Codes

The error code JXFS_E_PIN_FORMATINVALID is returned when:
- the subcommand in Byte 0 of msgData for Command secureMsgSend with protocol JXFS_PIN_PROTGENAS is not 01, 02 or 03.
- the subcommand in Byte 0 of msgData for Command secureMsgReceive with protocol JXFS_PIN_PROTGENAS is not 04.
- the subcommand in Byte 0 of msgData for Command secureMsgReceive with protocol JXFS_PIN_PROTGENAS is 01 and Byte 1 is not 00 and not 01 (Pin-Block format is not ISO-0 and ISO-1).
- the individual command data length for a subcommand is less than specified.

The error code JXFS_E_PIN_HSMSTATEINVALID is returned when:
- the subcommand in Byte 0 of msgData for Command secureMsgSend with protocol JXFS_PIN_PROTGENAS is 03 (Generate MAC) without a preceding GetMACRandom (secureMsgSend with subcommand 02).

The error code JXFS_E_PIN_MACINVALID is returned when
- the subcommand in Byte 0 of msgData for Command secureMsgReceive with protocol JXFS_PIN_PROTGENAS is 04 (Verify MAC) and the MACs didn't match.

The error code JXFS_E_PIN_KEYNOTFOUND is returned when
- the subcommand in Byte 0 of msgData for Command secureMsgSend with protocol JXFS_PIN_PROTGENAS is 01 (Generate PAC) and the device service doesn't find a master key.
- the subcommand in Byte 0 of msgData for Command secureMsgSend with protocol JXFS_PIN_PROTGENAS is 02 (Get MAC Random) and the device service doesn't find a master key.
- the subcommand in Byte 0 of msgData for Command secureMsgReceive with protocol JXFS_PIN_PROTGENAS is 04 (Verify MAC) and the device service doesn't find a key for the provided key generation and key version values.

The error code JXFS_E_PIN_NOPIN is returned when
- the subcommand in Byte 0 of msgData for Command secureMsgSend with protocol JXFS_PIN_PROTGENAS is 01 (Generate PAC) and no PIN or insufficient PIN-digits have been entered.

7.6.11 Command Sequence

The following list shows the sequence of actions an application has to take for the various GeldKarte Transactions. Please note that this is a summary and is just intended to clarify the purpose of the chipcard-related methods of the JxfsPINIso interface. In no way it can replace the ZKA specifications mentioned above.

Method	**protocol**	**message data**	**action of device service**
Preparation for Load/Unload			
secureMsgSend	CHIPZKA	Command APDU SELECT FILE DF_BÖRSE	
secureMsgReceive	CHIPZKA	Response APDU	recognize type of chip
secureMsgSend	CHIPZKA	Command APDU READ RECORD EF_ID	
secureMsgReceive	CHIPZKA	record EF_ID	store EF_ID

Method	**protocol**	**message data**	**action of device service**
secureMsgSend	CHIPZKA	Command APDU READ RECORD EF_LLOG	
secureMsgReceive	CHIPZKA	record EF_LLOG	
secureMsgSend	CHIPZKA	Command APDU READ_RECORD EF_BÖRSE	
secureMsgReceive	CHIPZKA	record EF_BÖRSE	
secureMsgSend	CHIPZKA	Command APDU READ_RECORD EF_BETRAG	
secureMsgReceive	CHIPZKA	record EF_BETRAG	
Load against other ec-Card			
secureMsgSend	CHIPZKA	**for type 0 chips only** Command APDU READ RECORD EF_KEYD	
secureMsgReceive	CHIPZKA	record EF_KEYD	
secureMsgSend	CHIPZKA	**for type 1 chips only** Command APDU GET KEYINFO	
secureMsgReceive	CHIPZKA	Response APDU	
secureMsgSend	CHIPZKA	Command APDU GET CHALLENGE	
secureMsgReceive	CHIPZKA	Random number RND1 from Chip	store RND1
secureMsgSend	CHIPZKA	Command APDU LADEN EINLEITEN with Secure Msg.	fill -Terminal ID -Traceno. -RND2 -MAC
secureMsgReceive	CHIPZKA	Response APDU	store response APDU for later check of ISOLZ message, BMP 62
secureMsgSend	ISOAZ	ISO8583 message 0200 Authorization Request	fill - Traceno. (BMP 11) - PAC (BMP 52) - RND_{MES} + RND_{PAC} (BMP 57) - MAC (BMP 64) check other security relevant fields
secureMsgReceive	ISOAZ	ISO8583 message 0210 Authorization Response	check MAC and other security relevant fields
secureMsgSend	ISOLZ	ISO8583 message 0200 Ladeanfrage	fill - Traceno. (BMP 11) - RND_{MES} (BMP 57) - MAC (BMP 64) check other security relevant fields.
secureMsgReceive	ISOLZ	ISO8583 message 0210 Ladeantwort	check MAC and other security relevant fields, store BMP62 for later use in LADEN command.
secureMsgSend	CHIPZKA	Command APDU GET CHALLENGE	
secureMsgReceive	CHIPZKA	Random number RND3 from chip	store RND3
secureMsgSend	CHIPZKA	Command APDU LADEN with Secure Msg.	provide complete command from BMP62 of ISOLZ response , compute command MAC
secureMsgReceive	CHIPZKA	Response APDU	check response MAC
GET_JOURNAL	ISOLZ	Vendor specific	
GET_JOURNAL	ISOAZ	Vendor specific	
Reversal of a Load against other ec-Card			
secureMsgSend	CHIPZKA	Command APDU SELECT FILE DF_BÖRSE	
secureMsgReceive	CHIPZKA	Response APDU	

Method	**protocol**	**message data**	**action of device service**
secureMsgSend	CHIPZKA	Command APDU GET CHALLENGE	
secureMsgReceive	CHIPZKA	Random number RND5 from chip	store RND5
secureMsgSend	CHIPZKA	Command APDU LADEN EINLEITEN with Secure Msg.	fill -Terminal ID -Traceno. -RND6 -Keyno. KGK_{LT} -MAC
secureMsgReceive	CHIPZKA	Response APDU	store response APDU for later check of ISOLZ message, BMP 62
secureMsgSend	ISOAZ	ISO8583 message 0400 Storno	fill - Traceno. (BMP 11) - PAC (BMP 52) - RND_{MES} + RND_{PAC} (BMP 57) - MAC (BMP 64) check other security relevant fields
secureMsgReceive	ISOAZ	ISO8583 message 0410 Storno Response	check MAC and other security relevant fields.
secureMsgSend	ISOLZ	ISO8583 message 0400 Storno	fill - Traceno. (BMP 11) - RND_{MES} (BMP 57) - MAC (BMP 64) check other security relevant fields.
secureMsgReceive	ISOLZ	ISO8583 message 0410 Storno Response	check MAC and other security relevant fields, store BMP62 for later use in LADEN command.
secureMsgSend	CHIPZKA	Command APDU GET CHALLENGE	
secureMsgReceive	CHIPZKA	Random number RND7 from chip	store RND7
secureMsgSend	CHIPZKA	Command APDU LADEN with Secure Msg.	provide complete command from BMP62 of ISOLZ response , compute command MAC
secureMsgReceive	CHIPZKA	Response APDU	check response MAC
GET_JOURNAL	ISOLZ	Vendor specific	
GET_JOURNAL	ISOAZ	Vendor specific	
PIN Verification Type 0			
secureMsgSend	CHIPZKA	Command APDU GET CHALLENGE	
secureMsgReceive	CHIPZKA	Random number RND0 from chip	store RND0
secureMsgSend	CHIPZKA	Command APDU EXTERNAL AUTHENTICATE	fill -Keyno. KINFO -ENCRND
secureMsgReceive	CHIPZKA	Response APDU	
secureMsgSend	CHIPZKA	Command APDU PUT DATA	fill RND1
secureMsgReceive	CHIPZKA	Response APDU	
secureMsgSend	CHIPZKA	Command APDU READ RECORD EF_INFO with Secure Messaging	
secureMsgReceive	CHIPZKA	record EF_INFO	check MAC
secureMsgSend	CHIPZKA	Command APDU GET CHALLENGE	
secureMsgReceive	CHIPZKA	Random number RND2 from chip	store RND2
secureMsgSend	CHIPZKA	Command APDU VERIFY	provide complete command APDU
secureMsgReceive	CHIPZKA	Response APDU	

Method	protocol	message data	action of device service
PIN Verification Type 1			
secureMsgSend	CHIPZKA	Command APDU GET KEYINFO	
secureMsgReceive	CHIPZKA	Response APDU	
secureMsgSend	CHIPZKA	Command APDU GET CHALLENGE	
secureMsgReceive	CHIPZKA	Random number RND0 from chip	store RND0
secureMsgSend	CHIPZKA	Command APDU MUTUAL AUTHENTICATE	fill ENC0
secureMsgReceive	CHIPZKA	Response APDU	check ENC1
secureMsgSend	CHIPZKA	Command APDU VERIFY	provide complete command APDU
secureMsgReceive	CHIPZKA	Response APDU	check MAC
„Laden vom Kartenkonto“ (both types)			
secureMsgSend	CHIPZKA	Command APDU LADEN EINLEITEN	fill -Terminal ID -Trace No.
secureMsgReceive	CHIPZKA	Response APDU	
secureMsgSend	ISOLZ	ISO8583 message 0200 Ladeanfrage	fill - Traceno. (BMP 11) - RNDMES (BMP 57) - MAC (BMP 64) check other security relevant fields.
secureMsgReceive	ISOLZ	ISO8583 message 0210 Ladeantwort	check MAC and other security relevant fields.
secureMsgSend	CHIPZKA	Command APDU LADEN	
secureMsgReceive	CHIPZKA	Response APDU	
GET_JOURNAL	ISOLZ	Vendor specific	
Reversal of a „Laden vom Kartenkonto“			
secureMsgSend	CHIPZKA	Command APDU SELECT FILE DF_BÖRSE	
secureMsgReceive	CHIPZKA	Response APDU	
secureMsgSend	CHIPZKA	Command APDU LADEN EINLEITEN	fill -Terminal ID -Traceno.
secureMsgReceive	CHIPZKA	Response APDU	
secureMsgSend	ISOLZ	ISO8583 message 0400 Storno	fill - Traceno. (BMP 11) - RNDMES (BMP 57) - MAC (BMP 64) check other security relevant fields.
secureMsgReceive	ISOLZ	ISO8583 message 0410 Storno Response	check MAC and other security relevant fields
secureMsgSend	CHIPZKA	Command APDU LADEN	
secureMsgReceive	CHIPZKA	Response APDU	
GET_JOURNAL	ISOLZ	Vendor specific	
Unload			

Method	**protocol**	**message data**	**action of device service**
secureMsgSend	CHIPZKA	ENTLADEN EINLEITEN	fill -Terminal ID -Trace No.
secureMsgReceive	CHIPZKA	Response APDU	
secureMsgSend	ISOLZ	ISO8583 message Entladeanfrage 0200	fill - Traceno. (BMP 11) - RNDMES (BMP 57) - MAC (BMP 64) check other security relevant fields.
secureMsgReceive	ISOLZ	ISO8583 message Entladeantwort 0210	check MAC and other security relevant fields
secureMsgSend	CHIPZKA	ENTLADEN	
secureMsgReceive	CHIPZKA	Response APDU	
secureMsgSend	CHIPZKA	ENTLADEN EINLEITEN	fill -Terminal ID -Trace No.
secureMsgReceive	CHIPZKA	Response APDU	
secureMsgSend	ISOLZ	ISO8583 message Entladequittung 0202	fill - Traceno. (BMP 11) - RNDMES (BMP 57) - MAC (BMP 64) check other security relevant fields.
secureMsgReceive	ISOLZ	ISO8583 message Entladebestätigung 0212	check MAC and other security relevant fields
secureMsgSend	CHIPZKA	Command APDU ENTLADEN	
secureMsgReceive	CHIPZKA	Response APDU	
GET_JOURNAL	ISOLZ	Vendor specific	
Repeated Messages (Stornowiederholung / Entladequittungswiederholung)			
secureMsgSend	ISOLZ	ISO8583 message Stronowiederholung 0401 or Entladebestätigungswiederholung 0203	fill - Traceno. (BMP 11) - RNDMES (BMP 57) - MAC (BMP 64) check other security relevant fields.
secureMsgReceive	ISOLZ	ISO8583 message Stornoantwort 410 or Entladequittung 0212	check MAC and other security relevant fields
GET_JOURNAL	ISOLZ	Vendor specific	

7.6.12 Command Sequence MAC/PAC non-ISO 8583

The following list shows sample sequence information of actions for handling non-ISO 8583 related MAC/PAC handling. Please note that this is a summary and is just intended to clarify the purpose of the chipcard-related methods of the JxfsPINIso interface. In no way it can replace the appropriate specifications.

Method	**protocol**	**message data**	**action of device service**
secureMsgSend	GENAS	Byte 0: 0x01 (Generate PAC) Byte 1: format (0 or 1) Byte 2-9: ANF (Primary Account Number, if length is less than 12 digits, value must be left padded with binary 0, only applicable for format	Generates a session key for PAC generation and finally the PAC itself. Determine generation and version values of Master-Key and return them along with the random value. OC data: Byte 0: key generation Byte 1: key version Byte 2-17: PAC random Byte 18-25: PAC value (all values are binary values)
secureMsgReceive	GENAS	Byte 0: 0x02 (Get MAC Random)	Generates a session key for MAC generation (see next step below). Determine generation and version values of Master-Key and return them along with the random value. OC data: Byte 0: key generation Byte 1: key version Byte 2-17: MAC random (all values are binary values)
secureMsgSend	GENAS	Byte 0: 0x03 (Generate MAC) Byte 1-n: Message to be mac'ed (all values are binary values)	Generates MAC over bytes 1-n of the inbound message using the session key created in the previous step. OC data: Byte 0-7: generated MAC(binary value).
secureMsgReceive	GENAS	Byte 0: 0x04 (Verify MAC) Byte 1: key generation Byte 2: key version Byte 3-18: MAC random Byte 19-26: MAC Byte 27-n: Message to be verified (all values are binary values) Note: If no message has been received, this function must be called by omitting Bytes 1-n	Generates a session key using the Master key identified by key generation and version by using the random value passed in. Generates a MAC for the message data passed in and compare the resulting MAC with the MAC passed in.

8 Appendix B: EMV Clarifications

EMV support by this specification consists in the ability of :

- importing Certification Authority and Chip Card rsa public keys,
- creating the PIN Blocks for offline PIN verification
- verifying static and dynamic authenticaton data

8.1 EMV Support

The PIN service is able to manage the EMV chip card regarding the card authentication and the RSA local PIN verification. Two steps are mandatory in order to reach these two functions

- The loading of the keys which come from the Certification Authorities or from the Chipcard itself
- AND the EMV PIN block format management

The Device Services is responsible for all key validation during the import process. The application is responsible for management of the key lifetime and expiry after the key is successfully imported

8.2 Key loading

The final goal of an application is to retrieve the keys located on a Chip card to perform the operations of authentication or local PIN check (RSA encrypted). These keys are provided by the card using EMV certificates and can be retrieved using a public key provided by a Certification Authority. The application should first load the keys issued by the Certification Authority. At transaction time the application will use these keys to load the keys that the application has retrieved from the chip card.

8.3 Certification Authority keys

These keys are provided in the following formats :

- Plain text
- Plain Text with EMV 2000 Verification Data
- EPI CA (or self signed) format as specified in the Europay International, EPI CA Module Technical – Interface specification Version 1.4
- PKCSV1_5 encrypted (as used by GIECB in France).

8.4 EPI CA format

The following table corresponds to table 4 of the Europay International, EPI CA Module Technical – Interface specification Version 1.4 and identifies the Europay Public Key (self-certified) and the associated data:

Field name	**Length**	**Description**	**Format**
ID of Certificate Subject	5	RID for Europay	Binary
Europay public key Index	1	Europay public key Index	Binary
Subject public key Algorithm Indicator	1	Algorithm to be used with the Europay public key Index, set to 0x01	Binary
Subject public key Length	1	Length of the Europay public key Modulus (equal to *Nca)*	Binary
Subject public key Exponent Length	1	Length of the Europay public key Exponent	Binary
Subject public key Exponent	1		Binary
Leftmost Digits of Subject public key	*Nca*-37	*Nca*-37 most significant bytes of the Europay public key Modulus	Binary
Subject public key Remainder	*37*	37 least significant bytes of the Europay public key Modulus	Binary
Subject public key Exponent	*1*	Exponent for Europay public key	Binary

Subject public key Certificate	*Nca*	Output of signature algorithm	Binary

Table 1

The following table corresponds to table 13 of the Europay International, EPI CA Module Technical – Interface specification Version 1.4 and identifies the Europay Public Key Hash code and associated data:

Field name	**Length**	**Description**	**Format**
ID of Certificate Subject	5	RID for Europay	Binary
Europay public key Index	1	Europay public key Index	Binary
Subject public key Algorithm Indicator	1	Algorithm to be used with the Europay public key Index, set to 0x01	Binary
Certification Authority public key Check Sum	20	Hash-code for Europay public key	Binary

Table 2

Table 2 corresponds to table 13 of the Europay International, EPI CA Module Technical – Interface specification Version 1.4

Chip card keys

These keys are provided as EMV certificates which come from the chip card in a multiple layer structure (issuer key first, then the ICC keys). Two kinds of algorithm are used with these certificates in order to retrieve the keys : One for the issuer key and the other for the ICC keys (ICC public key and ICC PIN encipherment key). The associated data with these algorithms – The PAN (Primary Account Number) and the SDA(Static Data to be Authenticated) - come also from the chip card

8.5 PIN block management

The PIN block management is done through the createPINBlock method. A new format JXFS_PIN_FMT_EMV has been added to indicate to the PIN service that the PIN block must follow the requirements of the EMVco, Book2 – Security & Key management Version 4.0 document The parameter *customerData* is used in this case to transfer to the PIN service the challenge number coming from the chip card. The final encryption must be done using a RSA public key. Please note that the application is responsible to send the PIN block to the chip card inside the right APDU

8.6 SHA-1 Digest

The SHA-1 Digest is a hash algorithm used by EMV in validating ICC static and dynamic data item. The SHA-1 Digest is supported through the computeSHA1Digest command. The application will pass the data to be hashed to the Device Service. Once the encryptor completes the SHA-1 hash code, the Device Service will return the 20-byte hash value back to the application.

9 Appendix C: Remote Key Loading Clarifications

9.1 Background Information

Most cryptographic functions used within Financial Industry transactions will continue to be based on symmetric key technology using either DES or triple DES. It is essential within symmetric key cryptography for the keys to be kept secret.

All the key exchanges between the host and the financial terminal are based on the initial encryption key (master key). This key was loaded at the installation of the self-service terminal (SST) and, usually, was the same during all the lifetime of the SST. The replacement of the master key needed human intervention and heavy safety rules in order to keep the master key secret and this key could not be downloaded because the current specifications do not provide all features required for supporting it in a branch or self-service environment for Remote Key Loading.

The new cryptographic rules in several countries mandate to replace the master key regularly.

This proposal provides mechanisms for the exchange of these symmetric keys in a secure automated way where the end points can be sure the data communicated is from a trusted source.

Remote Key loading allows an initial encryption symmetric key (master key) to be downloaded from a host using the Public Key Infrastructure (PKI) for encryption and verification of the master key.

The public key infrastructure (**PKI**) is based on asymmetric keys. An asymmetric key is composed of two parts:

- The **public key** part. It can be distributed to trusted persons.
- The **private key** part. It is kept secret by the one who generate it

This document is a proposal for 3 different mechanisms for remote key loading.

- **Remote key loading using signatures:** It is 2-parties Authentication scheme, where the host and SST authenticate each other directly. The key sent by the host is enciphered with RSA cryptographic method. The SST will decipher and verify the validity of the key before loading it into the security module.
- **Remote key loading using certificates:** It is a 3-parties Authentication scheme, where a third party, the certification authority, is ultimately responsible for the authentication / trust relationship.

9.2 Appendix C1: REMOTE KEY LOADING USING SIGNATURES

9.2.1 What is a digital signature?

A **digital signature** is a digital code that can be attached to an electronically transmitted message that uniquely identifies the sender. Like a written signature, the purpose of a digital signature is to guarantee that the individual sending the message really is who he or she claims to be.

9.2.2 How it works?

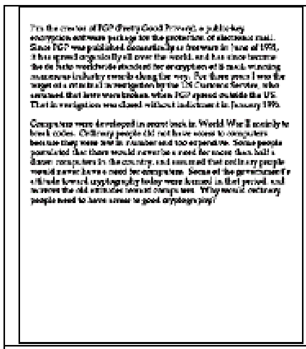
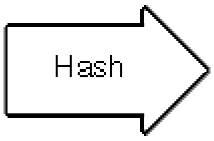
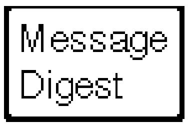
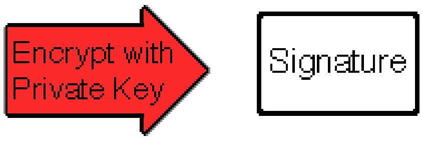
Digital signatures rely on a public key infrastructure (PKI). The PKI model involves an entity, such as a Host, having a pair of encryption keys – one private, one public. These keys work in consort to encrypt, decrypt and authenticate data. One way authentication occurs is through the application of a digital signature.

Private key: This is kept secret by the host

Public key: This key is distributed to trusted SST

For example:

- 1) The Host creates some data that it would like to digitally sign;
- 2) Host runs the data through a hashing algorithm to produce a hash or digest of the data. The digest is unique to every block of data – a digital fingerprint of the data, much smaller and therefore more economical to encrypt than the data itself;
- Encrypt the Digest the Host's private key. **This is the digital signature.**

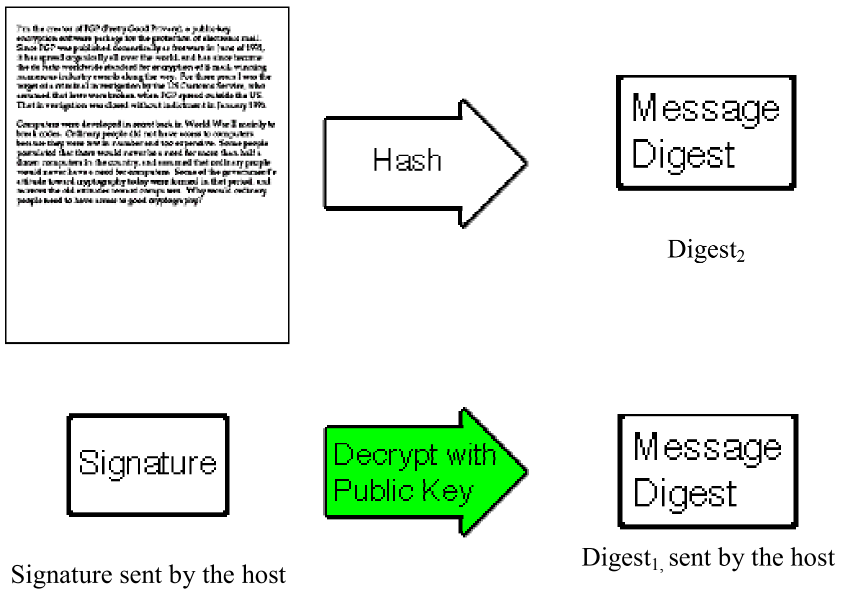
 I'm the creator of PGP (Pretty Good Privacy), a public-key encryption software package for the protection of electronic mail. Since PGP was published domestically as freeware in June of 1991, it has spread organically all over the world, and has since become the de facto worldwide standard for encryption of E-mail, winning numerous industry awards along the way. For three years I was the target of a criminal investigation by the US Customs Service, who assumed that laws were broken when PGP spread outside the US. That investigation was closed without indictment in January 1996. Computers were developed in secret back in World War II mainly to break codes. Ordinary people did not have access to computers because they were few in number and too expensive. Some people postulated that there would never be a need for more than half a dozen computers in the country, and assumed that ordinary people would never have a need for computers. Some of the government's attitude toward cryptography today were formed in that period, and mirrors the old attitudes toward computers. Why would ordinary people need to have access to good cryptography?			
1) Message to send to the host	2.1 Run hash algorithm	2.2 Message digest	2.3 Encrypt the message digest 5) signature

The Host then sends the following to the SST:

- data block;
- digital signature
- Host's public key

To validate the signature, the SST performs the following:

- SST runs data through the standard hashing algorithm – the same one used by the Host – to produce a digest of the data received. *Consider this digest$_2$*;
- SST uses the Host's public key to decrypt the digital signature. The digital signature was produced using the Host's private key to encrypt the data digest; therefore, when decrypted with the Host's public key it produces the same digest. *Consider this digest$_1$*. Incidentally, no other public key in the world would work to decrypt digest$_1$ – only the public key corresponding to the signing private key.
- SST compares digest$_1$ with digest$_2$



If digest$_1$ matches digest$_2$ exactly, the SST has confirmed the following:

- Data was not tampered with in transit. Changing a single bit in the data sent from the Host to the SST would cause digest$_2$ to be different than digest$_1$. Every data block has a unique digest; therefore, the SST detects an altered data block.
- Public key used to decrypt the digital signature corresponds to the private key used to create it. No other public key could possibly work to decrypt the digital signature, so the SST was not handed someone else's public key.

This gives an overview of Digital Signatures can be used in **Data Authentication**, in particular, to validate and securely install encryption keys.

9.2.3 Key Exchange using Digital Signatures

This section describes Key Exchange using Digital signatures.

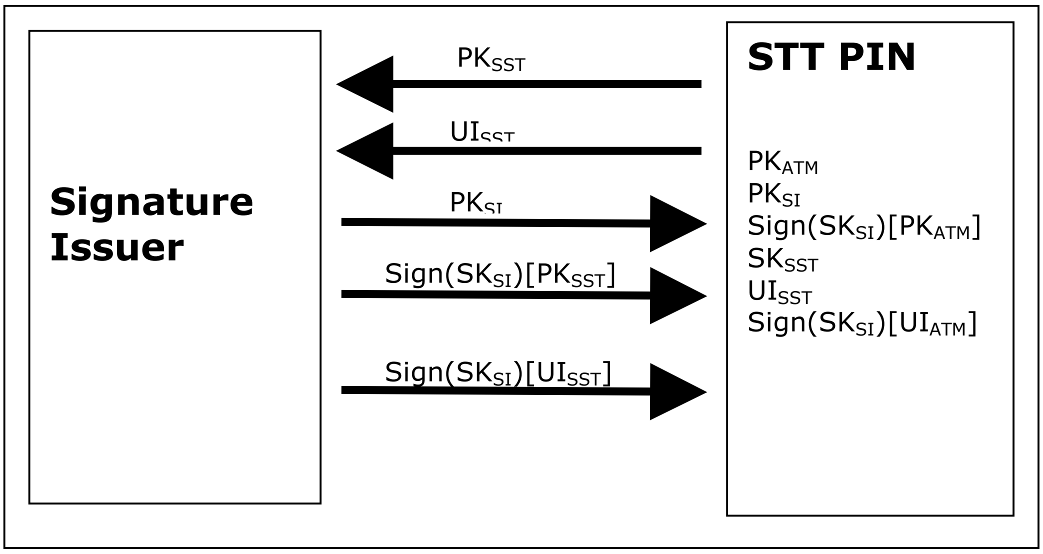
Initialization phase

At production time, a RSA key pair, which is unique for each device, is loaded into the PinPad.

Then a trusted third party, the Signature Issuer, is used to generate the signatures for the Public keys of each end point, ensuring their validity.

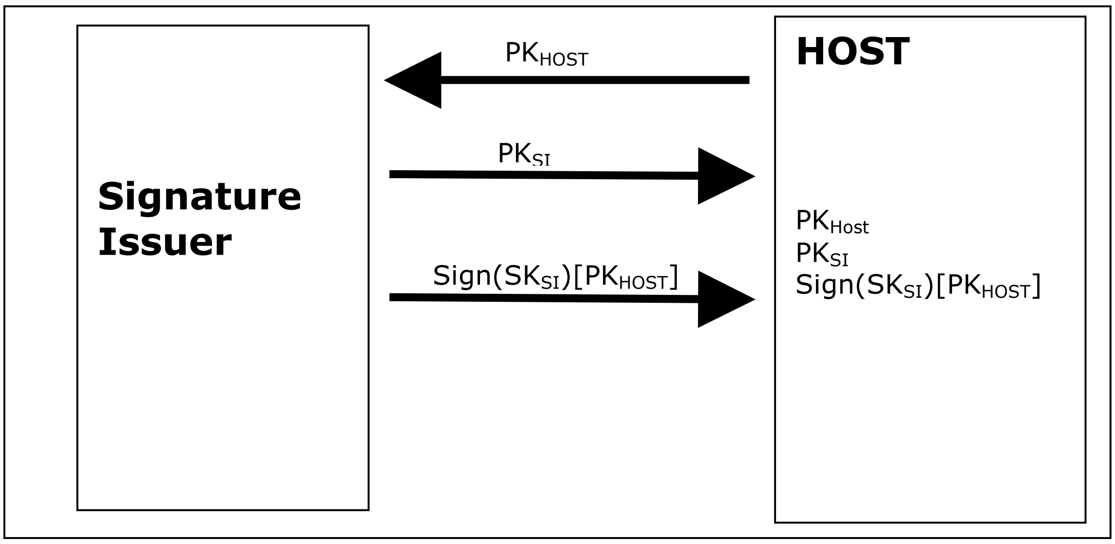
The initialization of the device is done in a secure environment; typically this is done during manufacture time or at the installation time in the customer premises.

Initialization Phase – Signature Issuer & SST PIN


Signature Issuer
STT PIN
PK_{SST}
UI_{SST}
PK_{SI}
$Sign(SK_{SI})[PK_{SST}]$
$Sign(SK_{SI})[UI_{SST}]$
PK_{ATM}
PK_{SI}
$Sign(SK_{SI})[PK_{ATM}]$
SK_{SST}
UI_{SST}
$Sign(SK_{SI})[UI_{ATM}]$

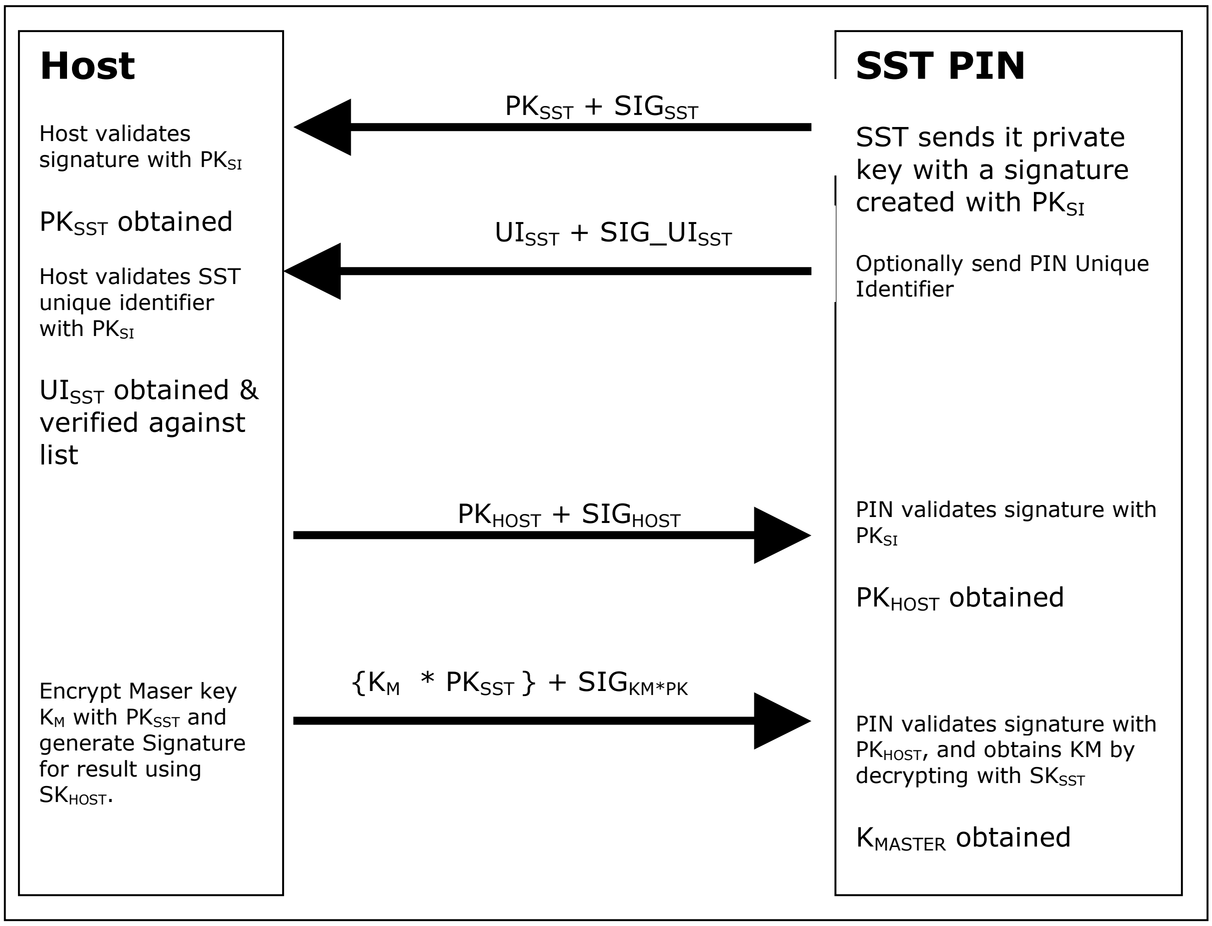
PK_{SST}	Public key of the Self Service Terminal. This key is either the one loaded in the PinPad in production or installation process or generated by the PinPad by the command *generateRSAKeyPair*.
UI_{SST}	Unique Identifier of the Self Service Terminal (Optional)
PK_{SI}	Public key of the Signature Issuer
$Sign(SK_{SI})[PK_{SST}]$	Signature of the Public key of the Self Service enciphered with the signature issuer public key.
$Sign(SK_{SI})[UI_{SST}]$	Signature of the unique identifier of the Self Service enciphered with the signature issuer public key.

Initialization Phase – Signature Issuer & HOST



PK_{HOST}	Public key of the Host
PK_{SI}	Public key of the Signature Issuer
$Sign(SK_{SI})[PK_{HOST}]$	Signature of the Public key of the Host enciphered with the signature issuer public key.

Key Exchange – Host & SST PIN


Host
Host validates signature with PK_SI
PK_SST obtained
Host validates SST unique identifier with PK_SI
UI_SST obtained & verified against list
Encrypt Maser key K_M with PK_SST and generate Signature for result using SK_HOST.
PK_SST + SIG_SST
UI_SST + SIG_UI_SST
PK_HOST + SIG_HOST
{K_M * PK_SST} + SIG_KM*PK
SST PIN
SST sends it private key with a signature created with PK_SI
Optionally send PIN Unique Identifier
PIN validates signature with PK_SI
PK_HOST obtained
PIN validates signature with PK_HOST, and obtains KM by decrypting with SK_SST
K_MASTER obtained

Step 1
The SST sends its Public Key to the Host in a secure structure: The SST PIN sends its SST Public Key with its associated Signature created by the Issuer 's Public Key. When the Host receives this information it will use the Signature Issuer's Public Key to validate the signature and obtain the SST Public Key.

Step 2 (Optional)
1. **The Host verifies that the key** it has just received is from a valid sender. It does this by obtaining the PIN device unique identifier. The SST PIN sends its Unique Identifier with its associated Signature created by the Issuer 's Public Key. When the Host receives this information it will use the Signature Issuer's Public Key to validate the signature and retrieve the PIN Unique Identifier. It can then check this against the list it received from the Signature Issuer. In a private SST network, it should not have any possibility of impersonation, i.e. that another device takes the role of an SST and fools the Host. The unique identifier prevents from any impersonation.

Step 3
The Host sends its public key to the SST: The Host sends its Public Key and associated Signature. The SST PIN verifies the signature using PK_{SI} and stores the key if it is valid.

Step 4
The SST PIN receives its Master Key from the Host: The Host encrypts the Master Key (K_M) with PK_{SST}. A signature for this is then created using SK_{HOST}. The SST PIN will then validate the signature using PK_{HOST} and then obtain the master key by decrypting using SK_{SST}.

9.3 Appendix C2: REMOTE KEY LOADING USING CERTIFICATES

9.3.1 What is a digital certificate?

Digital certificates are electronic files containing the user's public key and specific identifying information about the user. They are tamper-proof and cannot be forged. Much as a passport office does in issuing a passport, a Certification Authority certifies that the individual granted the digital certificate is who he or she claims to be.

Digital certificates do two things:

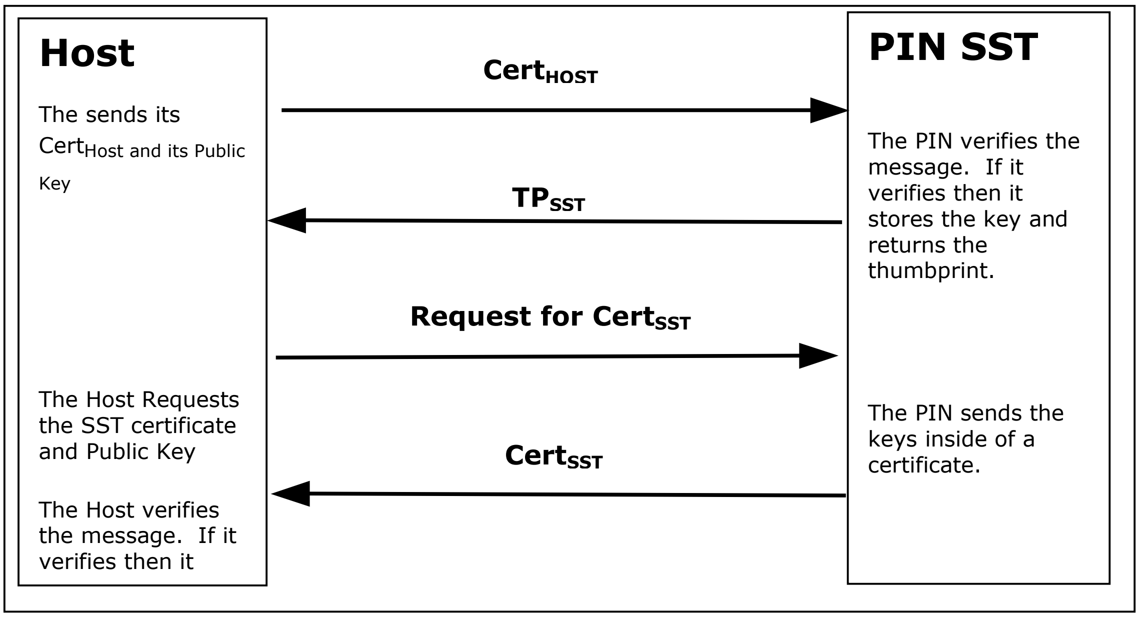
- They authenticate that their holders - people, web sites, and even network resources such as routers - are truly who or what they claim to be.
- They protect data exchanged online from theft or tampering.

9.3.2 What is a certification Authority?

A **Certification Authority** is a main component of a PKI. It is a trusted third party responsible for issuing digital certificates and managing them throughout their lifetime.
Certificate authorities (CA) are the digital world's equivalent of passport offices. They issue digital certificates and validate the holder's identity and authority. CA embed an individual's or an organization's public key along with other identifying information into each digital certificate and then cryptographically "sign" it as a tamper-proof seal, verifying the integrity of the data within it and validating its use.

9.3.3 Certificate Exchange and authentication

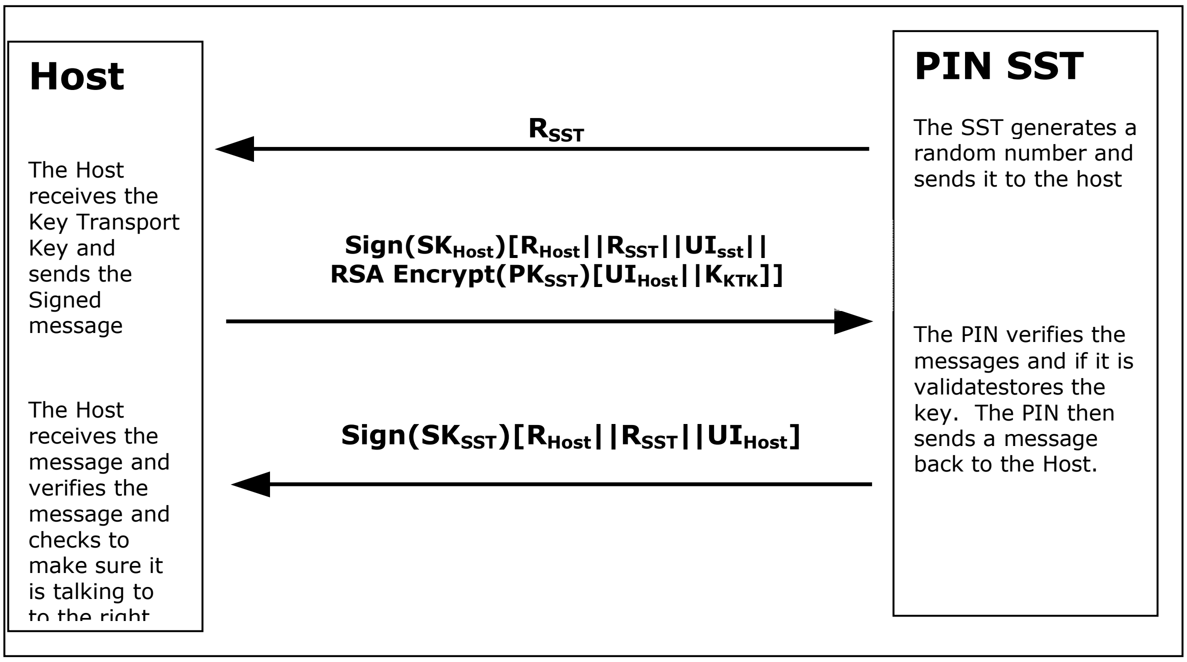
At this step the host and the SST exchange their certificates. The certificate contains the Public key of the sender.


Host
The sends its Cert$_{\text{Host and its Public Key}}$
The Host Requests the SST certificate and Public Key
The Host verifies the message. If it verifies then it
Cert$_{HOST}$
TP$_{SST}$
Request for Cert$_{SST}$
Cert$_{SST}$
PIN SST
The PIN verifies the message. If it verifies then it stores the key and returns the thumbprint.
The PIN sends the keys inside of a certificate.

Cert$_{HOST}$	Host certificate provided by a Certification Authority
TP$_{SST}$	SHA-1 Thumb print value returned by the *importCertificate* command.
Cert$_{SSt}$	SST certificate provided by a Certification Authority

Once the exchange of certificates is done the remote key loading can start.

9.3.4 Key Exchange – Host & SST PIN


Host
The Host receives the Key Transport Key and sends the Signed message
The Host receives the message and verifies the message and checks to make sure it is talking to to the right
R_{SST}
$Sign(SK_{Host})[R_{Host}||R_{SST}||UI_{sst}||$
$RSA\ Encrypt(PK_{SST})[UI_{Host}||K_{KTK}]]$
$Sign(SK_{SST})[R_{Host}||R_{SST}||UI_{Host}]$
PIN SST
The SST generates a random number and sends it to the host
The PIN verifies the messages and if it is validatestores the key. The PIN then sends a message back to the Host.

$\mathbf{R_{SST}}$	Random number generated by the SST encryptor
$\mathbf{SK_{Host}}$	Secret key of the host
$\mathbf{R_{Host}}$	Random number generated by the Host
$\mathbf{UI_{sst}}$	SST Unique identifier
$\mathbf{PK_{SST}}$	SST private key
$\mathbf{UI_{HOST}}$	HostUnique identifier
$\mathbf{K_{KTK}}$	Keys Transport Key

Step 1

The SST generates a random number and sends it to the host: The random number is unique for each key exchange process.

Step 2

The host constructs a key block data and sends it to the SST.

1) The host has obtained a Key Transport Key and wants to transfer it to the encryptor.
2) The host constructs a key block containing an identifier of the host, UI_{HOST}, the key, K_{KTK}, and enciphers the block, using the SST public key.
   RSA Encryption (PK_{SST})[UI_{HOST}] || K_{KTK}]

3) The host generates random data and builds the outer message containing the random number of the host, R_{HOST}, the random number of the SST, R_{SST}, and the SST unique Identifier, UI_{SST},

4) The host signs the whole block, containing sub steps 2 and 3 results, using its private key and sends the message to the SST.

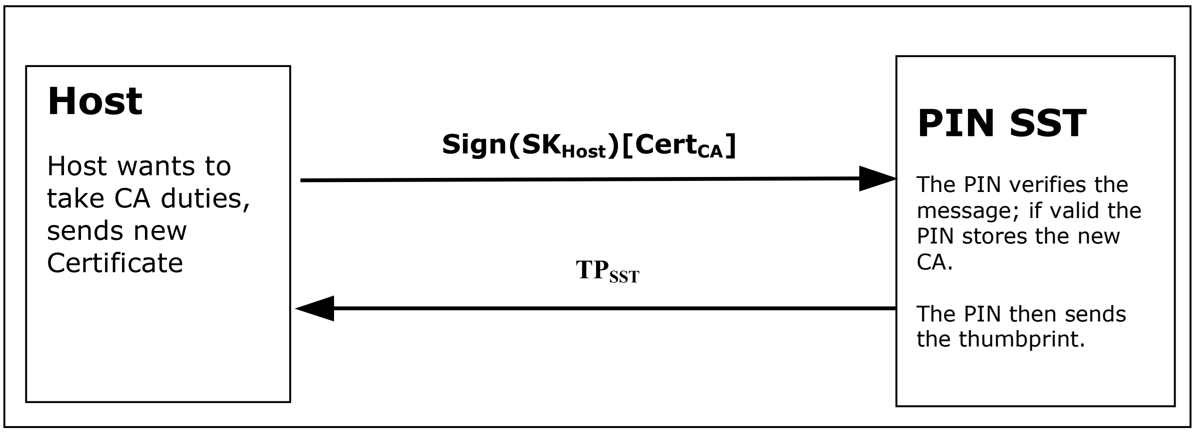
Step 3

The SST validates the key. The SST constructs a message that contains the host random number, R_{HOST}, the SST random number, R_{SST}, and the host identifier, UI_{HOST} , signed by the private signature of the SST and sends the message to the host.

9.3.5 Replace certificate

After the key is been loaded into the encryptor, the replacement if the loaded certificate could be completed (optional):

This is done by entity that would like to take over the job of being the CA. The new CA requests a Certificate from the previous Certificate Authority. The host must over-sign the message to take over the role of the CA to ensure that the HOST accepts the new Certificate Authority.


Host
Host wants to take CA duties, sends new Certificate
Sign(SK_Host)[Cert_CA]
TP_SST
PIN SST
The PIN verifies the message; if valid the PIN stores the new CA.
The PIN then sends the thumbprint.

Step 1

The HOST sends the message to the SST.

Step 2

The SST uses the host public key to verify the host signature. The SST uses the previous CA public key to verify the signature on the new Certificate sent by the host. If valid, the SST stores the new CA certificate and uses the new CA public key, as it's new CA verification key.

Step 3

The SST send to the host the thumb print value returned by the replace*Certificate* command

Primary Secondary certificate

Primary and Secondary Certificates for both the public verification key and public encipherment key are pre-loaded into the encryptor. Primary Certificates will be used until told otherwise by the host via the ***loadCertificate*** or ***replaceCertificate*** commands. The reason why the host would want to change states is because the HOST thinks that the Primary Certificates have been compromised.

After the host tells the encryptor to shift to the secondary certificate state, and then only Secondary Certificates can be used. The encryptor will no longer be able to go back to the Primary State and any attempts from the host to get or load a Primary Certificate will return an error.

MEMBERS :

DELARUE

DIEBOLD

DYNASTY

IBM

KAL

KEBA

LUTZ WOLF GRUPPE

NCR

NEXUS

SEIKO EPSON CORPORATION

WINCOR - NIXDORF

< End of Document >