



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

WORKSHOP AGREEMENT

CWA 13937-3

August 2000

ICS 35.240.40

J/eXtensions for Financial Services (J/XFS) for the Java Platform - Part
3: Magnetic Stripe & Chip Card Device Class Interface - Programmer's
Reference

This CEN Workshop Agreement can in no way be held as being an official standard
as developed by CEN National Members.

© 2000 CEN

All rights of exploitation in any form and by any means reserved world-wide for
CEN National Members

Ref. No CWA 13937-3:2000 E

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 1999-12-15/16 in Geneva and a subsequent electronic review by the Workshop participants, and the final version was sent to CEN for publication on 2000/06-21.

The specification is continuously reviewed and commented in the CEN/ISSS J/XFS Workshop. It is therefore expected that an update of the specification will be published in due time as a CWA, superseding this one. 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/iss/wkshop/j-xfs/cwa-updates>).

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 (iss@cenorm.be). To submit questions and comments for the J/XFS specifications, please contact the CEN/ISSS Secretariat (iss@cenorm.be) who will be forwarding them to the J/XFS Workshop.

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

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.

Contents

1	SCOPE.....	4
2	OVERVIEW	5
2.1	DESCRIPTION.....	5
2.2	CLASSES AND INTERFACES.....	7
2.3	SUPPORT CLASSES.....	8
3	DEVICE BEHAVIOR.....	9
3.1	DEVICE OPEN().....	9
4	CLASSES AND INTERFACES.....	10
4.1	ACCESS TO PROPERTIES.....	10
4.2	EXCEPTIONS	10
4.3	IJXFSMAGSTRIPECONTROL.....	11
4.4	IJXFSCHIPCARDCONTROL.....	15
4.5	IJXFSMOTORIZEDCARD	19
4.6	IJXFSMSDSECURE.....	23
5	SUPPORT CLASSES.....	27
5.1	JXFSMSDTRACKS	27
5.2	JXFSMSDTRACKSELECTION.....	29
5.3	JXFSMSDREADDATA	30
5.4	JXFSCCDDATA	32
5.5	JXFSMSDWMDATA.....	33
5.6	JXFSMSDSECUREMODE	34
5.7	JXFSMSDREADDATASECURE.....	35
6	CODES	37
6.1	ERROR CODES	37
6.2	STATUS CODES.....	37
6.3	OPERATION CODES.....	38
6.4	CONSTANTS.....	38

1 Scope

This document describes the Magnetic Stripe Device (MSD) as well as Chip Card Device (CCD) 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 Magnetic Stripe devices and Chip Card 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 document describes the J/XFS support classes for both Magnetic Stripe devices (MSD) as well as Chip Card devices (CCD).

As well as the rest of J/XFS device controls, J/XFS Magnetic Stripe and J/XFS Chip Card devices use the event driven model and the same behavioral model. Therefore, in the case of a Magnetic Stripe device, the application will instantiate a J/XFS Magnetic Stripe Device Control Object and then use the available methods to do I/O. When an I/O method is called, the J/XFS Magnetic Stripe 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 Magnetic Stripe Device Control Object for the various types of events it wishes to handle.

The same model applies to all J/XFS device controls and, in particular, to the Chip Card Device control.

2.1.1 Magnetic Stripe Device

The J/XFS Magnetic Stripe Reader/Encoder Device Support allows for the operation of devices with magnetic stripe read/write capabilities. Following are typical devices with such a capability:

- motor driven card reader/writer
- pull through card reader/writer
- dip card reader/writer

The following tracks and the corresponding international standards are taken into account in this document:

Track 1	ISO 7811
Track 2	ISO 7811
Track 3	ISO 7811 / ISO 4909

In addition to the pure reading of the tracks mentioned above, security boxes can be used via this service to check the data of writable tracks for manipulation. These boxes (such as CIM or MM) are sensor-equipped devices that are able to check some other information on the card and compare it with the track data.

Leds handling will be defined based on initialization configuration so no reference to them is made in this document.

Handling of *watermark* is also considered.

2.1.2 Chip Card Device

The J/XFS Chip Card Device Support allows for the operation of devices with chip access capabilities. Following are typical devices with such a capability :

- motor driven chip card devices.
- dip chip card devices.

The following chips and the corresponding international standards are taken into account in this document:

- Chip (contacted) ISO 7816

2.2 Classes and Interfaces

The following classes and interfaces are used by the J/XFS MSD and CCD Device Controls. In order to support the definition of the different properties of the different devices (see Introduction), the Device Controls are defined in a class hierarchy.

Class or Interface	Name	Description	Extends or Implements
Interface	IJxfsBaseControl	Base interface for all the device controls. Contains methods common to all the device controls.	
Interface	IJxfsMagStripeControl	Base interface for MSD controls. Contains method declarations specific to MSD controls.	Extends: IJxfsBaseControl
Interface	IJxfsChipCardControl	Base interface for CCD controls. Contains method declarations specific to CCD controls.	Extends: IJxfsBaseControl
Interface	IJxfsMotorizedCard	Interface for motorized card devices. Contains method declarations specific to motorized card devices.	
Interface	IJxfsMSDSecure	Interface for motorized card devices with secure module. Contains method declarations specific to card devices with secure module.	Extends: IJxfsMotorizedCard
Class	JxfsBaseControl	Base class for all the device controls. Contains properties common to all the device controls.	
Class	JxfsMagStripe	Base class for MSD controls. Contains properties specific to MSD device controls.	Implements: IjxfsMagStripeControl IJxfsMSDSecure
Class	JxfsChipCard	Base class for CCD controls. Contains properties specific to CCD device controls.	Implements: IjxfsChipCardControl IJxfsMotorizedCard

2.3 Support Classes

Class or Inter-face	Name	Description	Extends / Implements
Interface	JxfsConst	Interface containing the Jxfs constants that are common to several device categories	--
Interface	JxfsMSDConst	Interface containing the Jxfs constants that are common to all the MSD device controls.	--
Interface	JxfsCCDConst	Interface containing the Jxfs constants that are common to all the CCD device controls.	--
Interface	JxfsMotorizedCardConst	Interface containing the Jxfs constants for motorized card devices.	--
Class	JxfsMSDTracks	MSD Track selector class. Indicates for each track if its selected or not. Properties are read only.	Extends: JxfsType
Class	JxfsMSDTrackSelection	Subclass of MSD Track selector class. It contains the same properties but they can be set by applications.	Extends: JxfsMSDTracks
Class	JxfsMSDReadData	Data class that contains data returned in Operation Complete events for MSD <i>readData()</i> operation.	Extends: JxfsType
Class	JxfsCCDData	Data class that contains data returned in Operation Complete events for CCD input/output operations.	Extends: JxfsType
Class	JxfsMSDWmData	Data class that contains data returned in Operation Complete events for MSD <i>readWMtrack()</i> operation.	Extends: JxfsType
Class	JxfsMSDSecureMode	Data class that provides required properties for <i>readData()</i> operation in secure mode.	Extends: JxfsType
Class	JxfsMSDReadDataSecure	Data class that contains data returned in Operation Complete events for MSD <i>readData()</i> in secure mode.	Extends: JxfsType
Class	JxfsEvent	Abstract class from which all Jxfs event classes are extended	Extends: java.util.EventObject
Class	<i>Event</i> Event	The Device Service creates <i>Event</i> event instances of this class and delivers them through the J/XFS MSD Device Control's event callbacks to the application	Extends: JxfsEvent
Class	JxfsException	Exception class. The J/XFS MSD 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.
JXFS_E_OPEN	The open was already done by this Device Control.

4 Classes and Interfaces

All operation methods return an identificationID. If a method cannot be processed, 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.

getProperty

Syntax	Property <code>getProperty ()</code> throws <code>JxfsException</code>
Description	Returns the requested property.
Parameter	None
Event	No additional events are generated.
Exceptions	Some possible <code>JxfsException</code> <i>value codes</i> : <code>JXFS_E_CLOSED</code> <code>JXFS_E_UNREGISTERED</code> <code>JXFS_E_REMOTE</code>

setProperty

Syntax	void <code>setProperty (value)</code> throws <code>JxfsException</code>
Description	Sets the requested property.
Parameter	The desired property value.
Event	No additional events are generated
Exceptions	Some possible <code>JxfsException</code> <i>value codes</i> : <code>JXFS_E_CLOSED</code> <code>JXFS_E_UNREGISTERED</code> <code>JXFS_E_REMOTE</code> <code>JXFS_E_PARAMETER_INVALID</code>

4.2 Exceptions

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

Value	Meaning
<code>JXFS_E_CLOSED</code>	The Device Control has not been opened.
<code>JXFS_E_UNREGISTERED</code>	The device is not registered at the <code>JxfsDeviceManager</code> .
<code>JXFS_E_REMOTE</code>	A network error occurred.
<code>JXFS_E_CLAIMED</code>	The device is already claimed..
<code>JXFS_E_PARAMETER_INVALID</code>	A parameter is invalid.
<code>JXFS_E_NOT_SUPPORTED</code>	The function is not supported.

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

4.3 IJxfsMagStripeControl

4.3.1 Introduction

The J/XFS MSD Device Control Subclass is defined in JxfsMagStripe and is a subclass of JxfsBaseControl. Its interface is defined in IJxfsMagStripeControl interface which is a subclass of IJxfsBaseControl interface. The purpose of the J/XFS MSD 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.

Summary

Although IJxfsMagStripeControl 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 IJxfsMagStripeControl consists on the following methods:

- Getters of listed properties.
- Methods listed.

Property	Type	Access	Initialized after
deviceType	int	R	
mediaStatus	JxfsMediaStatus	R	
supportedReadTracks	JxfsMSDTracks	R	
supportedWriteTracks	JxfsMSDTracks	R	

Method	Return	May be used after
<i>getProperty</i>	<i>Property</i>	
readData	identificationID	
writeData	identificationID	

4.3.2 Properties

deviceType Property (R)

Type	<i>int</i>								
Initial Value	Depends on device type.								
Description	Identifies a type of MSD device. Depending on the device type it will be a combination of the following flags:								
	<table> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>JXFS_MSD_TYPE_SWIPE</td> <td>Swipe/pull through magnetic stripe reader/encoder.</td> </tr> <tr> <td>JXFS_MSD_TYPE_DIP</td> <td>Dip magnetic card reader/encoder.</td> </tr> <tr> <td>JXFS_MSD_TYPE_MOTOR</td> <td>Motorized card reader.</td> </tr> </tbody> </table>	Value	Meaning	JXFS_MSD_TYPE_SWIPE	Swipe/pull through magnetic stripe reader/encoder.	JXFS_MSD_TYPE_DIP	Dip magnetic card reader/encoder.	JXFS_MSD_TYPE_MOTOR	Motorized card reader.
Value	Meaning								
JXFS_MSD_TYPE_SWIPE	Swipe/pull through magnetic stripe reader/encoder.								
JXFS_MSD_TYPE_DIP	Dip magnetic card reader/encoder.								
JXFS_MSD_TYPE_MOTOR	Motorized card reader.								

mediaStatus Property (R)

Type	<i>JxfsMediaStatus</i>						
Initial Value	A JxfsMediaStatus (see related section in Base Architecture document).						
Description	Specifies the state of the media.						
Event	If the value of this property changes, the Device Service will send all registered StatusListeners a Status Event with the following values:						
	<table> <thead> <tr> <th>Field</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>status</td> <td>JXFS_S_MSD_MEDIA_STATUS <i>mediaStatus</i> has changed.</td> </tr> <tr> <td>details</td> <td>A <i>JxfsMediaStatus</i> object.</td> </tr> </tbody> </table>	Field	Value	status	JXFS_S_MSD_MEDIA_STATUS <i>mediaStatus</i> has changed.	details	A <i>JxfsMediaStatus</i> object.
Field	Value						
status	JXFS_S_MSD_MEDIA_STATUS <i>mediaStatus</i> has changed.						
details	A <i>JxfsMediaStatus</i> object.						

supportedReadTracks Property (R)

Type	<i>JxfsMSDTracks</i>
Initial Value	Null until open.
Description	Indicates which tracks can be physically read by the device.

supportedWriteTracks Property (R)

Type	<i>JxfsMSDTracks</i>
Initial Value	Null until open.
Description	Indicates which tracks can be physically written by the device.

4.3.3 Methods

readData Method

Syntax *identificationID readData (JxfsMSDTrackSelection tracksToRead) throws JxfsException;*

Description This method launches a read operation to obtain the data contained in the tracks specified by the *tracksToRead* parameter.

If media is present, the read operation is performed immediately. Otherwise, the device waits until it is present or the operation is cancelled.

After a successful completion of this input operation, an *OperationCompleteEvent* event is issued to inform the application of the results.

Parameter	Type	Name	Meaning
	<i>JxfsMSDTracksSelection</i>	<i>tracksToRead</i>	Tracks to be read.

Event **OperationCompleteEvent**
When a *readData ()* operation is completed an *OperationCompleteEvent* event will be sent by MSD Device Control to all registered *OperationCompleteListeners*. It will contain the data read.

Field	Value
<i>operationID</i>	JXFS_O_MSD_READDATA
<i>identificationID</i>	Identification ID of complete operation.
<i>result</i>	JXFS_RC_SUCCESSFUL Operation completed successfully.
	JXFS_E_CANCELLED Operation was cancelled.
	JXFS_E_MSD_READFAILURE No read conditions were satisfied (that is, not all tracks specified in <i>tracksToRead</i> parameter have been read).
	It is possible, however, that some tracks could be read. Check <i>data</i> object for extended information on tracks actually read.
	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 **JxfsMSDReadData** object.

IntermediateEvent

IntermediateEvent can be sent by MSD Device Control to all registered IntermediateListeners

Field	Value
<i>operationID</i>	JXFS_O_MSD_READDATA
<i>identificationID</i>	Identification ID of operation.
<i>reason</i>	JXFS_I_MSD_NO_MEDIA_PRESENT The read operation request cannot progress because there is no media inserted. JXFS_I_MSD_MEDIA_INSERTED The read operation request continues because a media has been inserted.
<i>data</i>	null

Exceptions

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

Value	Meaning
JXFS_E_MSD_NOTSUPPORT EDTRACK	At least one track specified in <i>tracksToRead</i> parameter is not supported by the device.
JXFS_E_MSD_NOTRACKS	No tracks specified in <i>tracksToRead</i> parameter.

writeData Method

Syntax *identificationID writeData (java.util.Vector wdata, boolean newCard) throws JxfsException;*

Description This method initiates a write operation of the data contained in *wdata*.

If media is present, the write operation is performed immediately. Otherwise, the device waits until it is present or the operation is cancelled. If the parameter *newCard* contains *true*, the card must be inserted *after* the operation is started.

Each vector element of *wdata* is a byte [] with the data to be written in each track. Vector element 0 contains data for track 1, vector element 1 contains data for track 2, and so on.

If no data has to be written for a given track, the corresponding vector element has to contain null.

After a successful completion of this output operation, an *OperationCompleteEvent* event is issued to inform the application of the results.

Parameter	Type	Name	Meaning
	java.util.Vector	wdata	Data to be written. Each vector element contains a byte [] of raw data per track. A null vector element is assumed no data to be written for its associated track.

boolean	newCard	If false, it specifies that the operation may proceed when a card is already present.
---------	---------	---

Event

OperationCompleteEvent

When a *writeData ()* operation is completed an *OperationCompleteEvent* event will be sent by MSD Device Control to all registered *OperationCompleteListeners*.

Field	Value
<i>operationID</i>	JXFS_O_MSD_WRITEDATA
<i>identificationID</i>	Identification Id of complete operation.
<i>result</i>	JXFS_RC_SUCCESSFUL Operation completed successfully.
	JXFS_E_CANCELLED Operation was cancelled by application or there was a card already present and <i>waitCard</i> value was true.
	JXFS_E_MSD_WRITEFAILURE No write conditions were satisfied. It is possible, however, that some tracks could be written. Check <i>data</i> object for extended information on tracks actually written.
	JXFS_E_MSD_NOMEDIA Media was removed before operation completion.
	JXFS_E_MSD_INVALIDMEDIA No appropriated media was found.
	JXFS_E_MSD_BADDATA Data is invalid.
	JXFS_E_MSD_MEDIAJAMMED Media is jammed.
	JXFS_E_MSD_SHUTTERFAIL Shutter could not be opened.
<i>data</i>	A JxfsMSDTracks object.

IntermediateEvent

IntermediateEvent can be sent by MSD Device Control to all registered *IntermediateListeners*

Field	Value
<i>operationID</i>	JXFS_O_MSD_WRITEDATA
<i>identificationID</i>	Identification Id of operation.
<i>reason</i>	JXFS_I_MSD_NO_MEDIA_PRESENT The write operation request cannot progress because there is no media inserted.
	JXFS_I_MSD_MEDIA_INSERTED The write operation request continues because a media has been inserted.

<i>data</i>	null
-------------	------

Exceptions

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

Value	Meaning
JXFS_E_MSD_NOTSUPPORT EDTRACK	At least one of the specified tracks is not supported by the device.
JXFS_E_MSD_NOTRACKS	No track data has been specified.

4.4 JxfsChipCardControl

4.4.1 Introduction

The J/XFS Chip Card Device Control Subclass is defined in JxfsChipCard and is a subclass of JxfsDeviceControl. Its interface is defined in IJxfsCCDControl interface which is a subclass of IJxfsBaseControl interface. The purpose of the J/XFS CCD 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.

This class represents a physical device (or part of it) that has chip card access capabilities (send/receive of commands and data).

Summary

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

Therefore, the IJxfsChipCardControl consists on the following methods:

- Getters of listed properties.
- Methods listed.

Property	Type	Access	Initialized after
deviceType	int	R	
mediaStatus	JxfsMediaStatus	R	

Method	Return	May be used after
getProperty	Property	
isCcdT	boolean	
chipInit	identificationID	
chipIO	identificationID	

4.4.2 Properties

deviceType Property (R)

Type	<i>int</i>												
Initial Value	Depends on device type.												
Description	Identifies a type of Chip Card device. Depending on the device type it will be a combination of the following flags:												
	<table> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>JXFS_CCD_TYPE_SWIPE</td> <td>Swipe/pull through chip card device.</td> </tr> <tr> <td>JXFS_CCD_TYPE_DIP</td> <td>Dip chip card device.</td> </tr> <tr> <td>JXFS_CCD_TYPE_MOTOR</td> <td>Motorized chip card device.</td> </tr> <tr> <td>JXFS_CCD_TYPE_CONTACT</td> <td>Contactless chip card device.</td> </tr> <tr> <td>LESS</td> <td></td> </tr> </tbody> </table>	Value	Meaning	JXFS_CCD_TYPE_SWIPE	Swipe/pull through chip card device.	JXFS_CCD_TYPE_DIP	Dip chip card device.	JXFS_CCD_TYPE_MOTOR	Motorized chip card device.	JXFS_CCD_TYPE_CONTACT	Contactless chip card device.	LESS	
Value	Meaning												
JXFS_CCD_TYPE_SWIPE	Swipe/pull through chip card device.												
JXFS_CCD_TYPE_DIP	Dip chip card device.												
JXFS_CCD_TYPE_MOTOR	Motorized chip card device.												
JXFS_CCD_TYPE_CONTACT	Contactless chip card device.												
LESS													

mediaStatus Property (R)

Type	<i>JxfsMediaStatus</i>
Initial Value	A JxfsMediaStatus (see related section in Base Architecture document).
Description	Specifies the state of the media.
Event	If the value of this property changes, the Device Service will send all registered StatusListeners a Status Event with one of the following

values:	
Field	Value
status	JXFS_S_CCD_MEDIA_STATUS <i>mediaStatus</i> has changed.
details	A <i>JxfsMediaStatus</i> object.

4.4.3 Methods

isCcdT Method

Syntax	<i>boolean isCcdT (int noOfProtocol) throws JxfsException;</i>		
Description	This method is used to obtain information on which protocols are supported by the device. Returns TRUE if protocol Tnn, where nn is the value of the parameter, is supported, FALSE otherwise.		
Parameter	Type	Name	Meaning
	int	noOfProtocol	Number of protocol being queried, from 0 to 15 for protocols T0 to T15.
Exceptions	No additional exceptions are generated. See section on JxfsExceptions for common value codes.		

chipInit Method

Syntax	<i>identificationID chipInit () throws JxfsException;</i>		
Description	Performs a chip card initialization and reads the answer to reset (ATR) data. If media is present, the operation is performed immediately. Otherwise, the device waits until it is present or the operation is cancelled . After a successful completion of this operation, an <i>OperationCompleteEvent</i> event is issued to inform the application of the result.		
Event	OperationCompleteEvent When a <i>chipInit()</i> operation is completed an <i>OperationCompleteEvent</i> event will be sent by CCD Device Control to all registered <i>OperationCompleteListeners</i> . It will contain the data read.		
	Field	Value	
	<i>operationID</i>	JXFS_O_CCD_CHIPINIT	
	<i>identificationID</i>	Identification Id of complete operation.	
	<i>result</i>	JXFS_RC_SUCCESSFUL	Operation completed successfully. Check <i>data</i> field for ATR data from chip.
		JXFS_E_CANCELLED	Operation was cancelled.
		JXFS_E_CCD_IOERROR	IO error occurred. No ATR data is available.
		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

data JXFS_E_CCD_SHUTTERFAIL
Shutter could not be opened.
A **JxfsCCDData** object.
It contains ATR data from chip.

IntermediateEvent

IntermediateEvent can be sent by CCD Device Control to all registered IntermediateListeners

Field	Value
<i>operationID</i>	JXFS_O_CCD_CHIPINIT
<i>identificationID</i>	Identification Id of operation.
<i>reason</i>	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.
<i>data</i>	null

Exceptions No additional exceptions are generated. See section on JxfsExceptions for common value codes.

chipIO Method

Syntax *identificationID chipIO (byte[] chipData, int protocol) throws JxfsException;*

Description This method initiates an input/output operation. The contents of *chipData* is sent to the chip card. Replied data from the chip card is returned to the application in an *OperationCompleteEvent* event. The parameter *protocol* specifies the protocol to use.

After a successful completion of this operation, a *OperationCompleteEvent* event is issued to inform the application of the results.

Parameter	Type	Name	Meaning
	byte[]	chipData	Data to be sent.
	int	protocol	Protocol to be used (0..15).

Event **OperationCompleteEvent**

When a *chipIO ()* operation is completed an *OperationCompleteEvent* event will be sent by CCD Device Control to all registered *OperationCompleteListeners*. It will contain the data read.

Field	Value
<i>operationID</i>	JXFS_O_CCD_CHIPIO
<i>identificationID</i>	Identification Id of complete operation.
<i>result</i>	JXFS_RC_SUCCESSFUL Operation completed successfully. Check <i>data</i> field for data returned from chip. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_CCD_IOERROR IO error occurred. No data is available. 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.

data JXFS_E_CCD_BADDATA
Chip reported data was bad.
JXFS_E_CCD_BADPROTOCOL
Protocol not supported.
A **JxfsCCDData** object.
It contains data returned from chip if operation completed successfully.

IntermediateEvent

IntermediateEvent can be sent by CCD Device Control to all registered IntermediateListeners

Field	Value
<i>operationID</i>	JXFS_O_CCD_CHIPIO
<i>identificationID</i>	Identification Id of operation.
<i>reason:</i>	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.
<i>data</i>	null

Exceptions

No additional exceptions are generated. See section on JxfsExceptions for common value codes.

4.5 IJxfsMotorizedCard

4.5.1 Introduction

This interface contains those properties and functions commonly supported in motorized card devices (such as motorized magnetic card readers/encoder and chip card stations) related with its mechanical capabilities like eject or retain cards.

It is intended that this interface will be implemented by device controls that represent physical devices able to manage cards with chip or magnetic stripes (that is, subclasses of JxfsMagStripe and JxfsChipCard classes) that are equipped with motorized and mechanical capabilities.

Summary

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

Therefore, the IJxfsMotorizedCard consists on the following methods:

- Getters of listed properties.
- Methods listed.

Property	Type	Access	Initialized after
powerOffCapabilities	int	R	
powerOnCapabilities	int	R	
retainBinStatus	JxfsThresholdStatus	R	
retainCardCount	int	R	
retainCapability	boolean	R	
secureModuleType	int	R	

Method	Return	May use after
<i>getProperty</i>	<i>Property</i>	
<i>setProperty</i>	<i>void</i>	
resetRetainCardCount	<i>void</i>	
ejectCard	identificationID	
retainCard	identificationID	

4.5.2 Properties

powerOffCapabilities Property (R)

Type	<i>int</i>
Initial Value	Depends on device.
Description	Indicates the action taken by the device at power off if media is present. Depending on the device capabilities it will be set with one of the following values:
Value	Meaning
JXFS_MOTOR_EJECT	Card is ejected.
JXFS_MOTOR_EJECT_THEN_RETAIN	Card is ejected, then, after some seconds, it is retained.
JXFS_MOTOR_NOACTION	No action is taken.
JXFS_MOTOR_READ_POSITION	Card is brought to the read/write position.
JXFS_MOTOR_RETAIN	Card is retained.

powerOnCapabilities Property (R)

Type	<i>int</i>
-------------	------------

Initial Value	Depends on device.
Description	Indicates the action taken by the device at power on if media is present. Depending on the device capabilities it will be set with one of the following values:
Value	Meaning
JXFS_MOTOR_EJECT	Card is ejected.
JXFS_MOTOR_EJECT_THEN_RETAIN	Card is ejected, then, after some seconds, it is retained.
JXFS_MOTOR_NOACTION	No action is taken.
JXFS_MOTOR_READ_POSITION	Card is brought to the read/write position.
JXFS_MOTOR_RETAIN	Card is retained.

retainBinStatus Property (R)

Type	<i>JxfsThresholdStatus</i>
Initial Value	A <i>JxfsThresholdStatus</i> (see related section in Base Architecture document).
Description	Indicates the fill status of the retain bin, if supported.
Event	If the value of this property changes, the Device Service will send all registered StatusListeners a Status Event with the following value:
Field	Value
status	JXFS_S_MOTOR_BIN_STATUS <i>retainBinStatus</i> has changed.
details	A <i>JxfsThresholdStatus</i> object.

retainCardCount Property (R/W)

Type	<i>int</i>
Initial Value	Depends on device at open.
Description	Number of cards retained. This value is persistent between power failures, open and close. The <i>resetRetainCardCount</i> method resets this property to 0.
Event	If the value of this property changes (increments), the Device Service will send all registered StatusListeners a StatusEvent with a status value of:
Field	Value
status	JXFS_S_MOTOR_BIN_CARDRETAINED <i>retainCardCount</i> has incremented.
details	None.

retainCapability Property (R)

Type	<i>boolean</i>
Initial Value	Depends on device.
Description	Indicates if device is able to retain cards. True means it is able to retain, false no retain capability support.

secureModuleType Property (R)

Type	<i>int</i>
Initial Value	Depends on device.
Description	Contains the secure module type, if any being used by the device.
Value	Meaning

JXFS_MSD_SECTYPE_NOTSU No security module available.
 PPORTED
 JXFS_MSD_SECTYPE_MMBO MMBBox module.
 X
 JXFS_MSD_SECTYPE_CIM86 CIM86 module

4.5.3 Methods

resetRetainCardCount Method

Syntax *void resetRetainCardCount ()*
Description Sets *retainCardCount* property to 0.

ejectCard Method

Syntax *identificationID ejectCard () throws JxfsException;*

Description Ejects the card allowing card taking from user.

Event

OperationCompleteEvent

When a *ejectCard()* operation is completed an *OperationCompleteEvent* event will be sent by the Device Control to all registered *OperationCompleteListener*s with the following data:

Field	Value
<i>operationID</i>	JXFS_O_MOTOR_EJECTCARD
<i>identificationID</i>	The corresponding Id.
<i>result</i>	JXFS_RC_SUCCESSFUL Operation completed successfully. This implies that the media has been presented.
	JXFS_E_CANCELLED Operation was cancelled.
	JXFS_E_MOTOR_MEDIAJAMMED Media is jammed.
	JXFS_E_MOTOR_SHUTTERFAIL Shutter could not be opened.
	JXFS_E_MOTOR_NOMEDIA There is no media to eject.
<i>data:</i>	null.

IntermediateEvent

IntermediateEvent can be sent by CCD Device Control to all registered *IntermediateListeners*

Field	Value
<i>operationID</i>	JXFS_O_MOTOR_EJECTCARD
<i>identificationID</i>	Identification Id of operation.
<i>result:</i>	JXFS_I_MOTOR_MEDIATAKEN The media has been taken by the user.
<i>data</i>	null

Exceptions No additional exceptions are generated. See section on *JxfsExceptions* for common value codes.

retainCard Method

Syntax	<i>identificationID retainCard () throws JxfsException;</i>								
Description	Retains card.								
Event	OperationCompleteEvent When a <i>retainCard()</i> operation is completed an <i>OperationCompleteEvent</i> event will be sent by the Device Control to all registered <i>OperationCompleteListener</i> s. <table><thead><tr><th>Field</th><th>Value</th></tr></thead><tbody><tr><td><i>OperationID</i></td><td>JXFS_O_MOTOR_RETAINCARD</td></tr><tr><td><i>IdentificationID</i></td><td>The corresponding Id.</td></tr><tr><td><i>Result</i></td><td>JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_MOTOR_BINFULL Retain bin is full. JXFS_E_MOTOR_MEDIAJAMMED Media is jammed. JXFS_E_MOTOR_NOMEDIA There is no media to retain.</td></tr></tbody></table>	Field	Value	<i>OperationID</i>	JXFS_O_MOTOR_RETAINCARD	<i>IdentificationID</i>	The corresponding Id.	<i>Result</i>	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_MOTOR_BINFULL Retain bin is full. JXFS_E_MOTOR_MEDIAJAMMED Media is jammed. JXFS_E_MOTOR_NOMEDIA There is no media to retain.
Field	Value								
<i>OperationID</i>	JXFS_O_MOTOR_RETAINCARD								
<i>IdentificationID</i>	The corresponding Id.								
<i>Result</i>	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. JXFS_E_MOTOR_BINFULL Retain bin is full. JXFS_E_MOTOR_MEDIAJAMMED Media is jammed. JXFS_E_MOTOR_NOMEDIA There is no media to retain.								
Exceptions	<i>data</i> null No additional exceptions are generated. See section on <i>JxfsExceptions</i> for common value codes.								

4.6 IJxfsMSDSecure

4.6.1 Introduction

This interface contains properties and functions that may be supported in motorized card MSD devices with a security box installed.

It is intended that this interface will be implemented by device controls that represent physical devices with the security feature..

Summary

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

Therefore, the IJxfsMSDSecure consists on the following methods:

- Getters of listed properties.
- Methods listed.

Property	Type	Access	Initialized after
secureModuleKey	byte []	R/W	
secureModuleStatus	int	R	

Method	Return	May be used after
<i>getProperty</i>	<i>Property</i>	
<i>setProperty</i>	<i>void</i>	
readData	identificationID	
readWMtrack	identificationID	

4.6.2 Properties

secureModuleKey Property (R/W)

Type	<i>byte []</i>
Initial Value	null
Description	Contains the secure module key with parity. Its value should be introduced once and be kept after power off.

secureModuleStatus Property (R)

Type	<i>int</i>								
Initial Value	Depends on device at open.								
Description	Indicates the status of the security module, if any.								
	<table> <thead> <tr> <th>Value</th> <th>Meaning</th> </tr> </thead> <tbody> <tr> <td>JXFS_S_MSD_SEC_READY</td> <td>Security module ready.</td> </tr> <tr> <td>JXFS_S_MSD_SEC_NOTREADY</td> <td>Security module not ready.</td> </tr> <tr> <td>JXFS_S_MSD_SEC_UNKNOWN</td> <td>State of the security module cannot be determined with the device in its current state.</td> </tr> </tbody> </table>	Value	Meaning	JXFS_S_MSD_SEC_READY	Security module ready.	JXFS_S_MSD_SEC_NOTREADY	Security module not ready.	JXFS_S_MSD_SEC_UNKNOWN	State of the security module cannot be determined with the device in its current state.
Value	Meaning								
JXFS_S_MSD_SEC_READY	Security module ready.								
JXFS_S_MSD_SEC_NOTREADY	Security module not ready.								
JXFS_S_MSD_SEC_UNKNOWN	State of the security module cannot be determined with the device in its current state.								
Event	<p>If the value of this property changes, the Device Service will send all registered StatusListeners a StatusEvent with a status value of:</p> <table> <thead> <tr> <th>Field</th> <th>Value</th> </tr> </thead> <tbody> <tr> <td>status</td> <td>JXFS_S_MSD_SEC_STATUS</td> </tr> <tr> <td>details</td> <td><i>secureModuleStatus</i> has changed. None.</td> </tr> </tbody> </table>	Field	Value	status	JXFS_S_MSD_SEC_STATUS	details	<i>secureModuleStatus</i> has changed. None.		
Field	Value								
status	JXFS_S_MSD_SEC_STATUS								
details	<i>secureModuleStatus</i> has changed. None.								

4.6.3 Methods

readData Method

Syntax *identificationID readData (JxfsMSDTrackSelection tracksToRead, JxfsMSDSecureMode secureMode) throws JxfsException;*

Description This method overloads the normal readData method.

It launches a read operation to obtain the data contained in the tracks specified by the *tracksToRead* parameter.

If media is present, the read operation is performed immediately. Otherwise, the device waits until it is present or the operation is cancelled.

After a successful completion of this input operation, an *OperationCompleteEvent* event is issued to inform the application of the results.

Parameter	Type	Name	Meaning
	JxfsMSDTracksSelection	tracksToRead	Tracks to be read.
	JxfsMSDSecureMode	secureMode	Required settings for secure operation.

Event **OperationCompleteEvent**
When a *readData ()* operation is completed an *OperationCompleteEvent* event will be sent by MSD Device Control to all registered *OperationCompleteListeners*. It will contain the data read.

Field	Value
<i>operationID</i>	JXFS_O_MSD_READDATA
<i>identificationID</i>	Identification ID of complete operation.
<i>result</i>	JXFS_RC_SUCCESSFUL Operation completed successfully.
	JXFS_E_CANCELLED Operation was cancelled.
	JXFS_E_MSD_READFAILURE No read conditions were satisfied (that is, not all tracks specified in <i>tracksToRead</i> parameter have been read). It is possible, however, that some tracks could be read. Check <i>data</i> object for extended information on tracks actually read.
	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.
<i>data</i>	A JxfsMSDReadDataSecure object.

IntermediateEvent
IntermediateEvent can be sent by MSD Device Control to all registered *IntermediateListeners*

Field	Value
<i>operationID</i>	JXFS_O_MSD_READDATA
<i>identificationID</i>	Identification ID of operation.

<i>reason</i>	JXFS_I_MSD_NO_MEDIA_PRESENT The read operation request cannot progress because there is no media inserted. JXFS_I_MSD_MEDIA_INSERTED The read operation request continues because a media has been inserted.
<i>data</i>	null

Exceptions

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

Value	Meaning
JXFS_E_MSD_NOTSUPPORT EDTRACK	At least one track specified in <i>tracksToRead</i> parameter is not supported by the device.
JXFS_E_MSD_NOTRACKS	No tracks specified in <i>tracksToRead</i> parameter.
JXFS_E_MSD_NOTSUPPORT EDCAP	The service does not have secure capability.

readWMtrack

Syntax

identificationID readWMtrack () throws JxfsException;

Description

This method launches a read operation to obtain the data contained in the Watermark.

If media is present, the read operation is performed immediately. Otherwise, the device waits until it is present or the operation is cancelled.

After a successful completion of this input operation, an *OperationCompleteEvent* event is issued to inform the application of the results.

Event

OperationCompleteEvent

When a *readData ()* operation is completed an *OperationCompleteEvent* event will be sent by MSD Device Control to all registered *OperationCompleteListeners*. It will contain the data read.

Field	Value
<i>operationID</i>	JXFS_O_MSD_READDATA
<i>identificationID</i>	Identification ID of complete operation.
<i>result</i>	JXFS_RC_SUCCESSFUL Operation completed successfully. JXFS_E_CANCELLED Operation was cancelled. 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.

Exceptions

data A **JxfsMSDWmData** with Watermark data.
Some possible *JxfsException value codes*. See section on *JxfsExceptions* for other *JxfsException value codes*.

Value	Meaning
--------------	----------------

JXFS_E_MSD_NOTSUPPORT Watermark is not supported.
EDTRACK

5 Support Classes

5.1 JxfsMSDTracks

This class provides properties and methods to query which tracks of a MSD device have been selected, are active or have been written.

Used by readData method.

Summary

Implements : --

Extends : JxfsType

Property	Type	Access	Initialized after
track1	boolean	R	
track2	boolean	R	
track3	boolean	R	

Method	Return	May use after
<i>isProperty</i>	<i>Property</i>	
allTracks	boolean	
noTracks	boolean	
JxfsMSDTracks (boolean track1, boolean track2, boolean track3)	(constructor of the class)	

5.1.1 Properties

track1 Property (R)

Type	<i>boolean</i>	
Initial Value	FALSE	
Description	Indicates if track1 is selected.	
Value	FALSE	Meaning
	TRUE	Track1 is not selected.
		Track1 is selected.

track2 Property (R)

Type	<i>boolean</i>	
Initial Value	FALSE	
Description	Indicates if track2 is selected.	
Value	FALSE	Meaning
	TRUE	Track2 is not selected.
		Track2 is selected.

track3 Property (R)

Type	<i>boolean</i>	
Initial Value	FALSE	
Description	Indicates if track3 is selected.	
Value	FALSE	Meaning
	TRUE	Track3 is not selected.
		Track3 is selected.

5.1.2 Methods

isTrack1 .. isTrack3 Methods

Syntax
Description

boolean isTrack1 () .. boolean isTrack3 ()
Return TRUE if specific track property is set to TRUE.

allTracks Method

Syntax
Description

boolean allTracks ()
Returns TRUE if all tracks (*track1*, *track2* and *track3*) are set to TRUE.

noTracks Method

Syntax
Description

boolean noTracks ()
Returns TRUE if all tracks (*track1*, *track2* and *track3*) are set to FALSE.

JxfsMSDTracks Constructor

Syntax
Description

JxfsMSDTracks (boolean track1, boolean track2, boolean track3)
Constructor of the class.

JXFS_E_MSD_NOTSUPPORTED	Track not supported by device.
DTRACK	
JXFS_E_MSD_READFAILURE	Read error on track.
JXFS_E_MSD_PARITY	Parity read error.
JXFS_E_MSD_READ_EOF	Only SS,SE,BCC on track.
JXFS_E_MSD_NO_STRIPE	No magnetic stripe or flux on stripe detected (if device has capability to detect this situation).
JXFS_E_MSD_READ_OTHER	Any other type of error.

5.3.2 Methods

JxfsMSDReadData Constructor

Syntax

JxfsMSDReadData (java.util.Vector dataRead, JxfsMSDTracks tracksRead, int resultReadTrack1, int resultReadTrack2, int resultReadTrack3)

Description

Constructor of the class.

5.7.2 Methods

JxfsMSDReadDataSecure Constructor

Syntax	<i>JxfsMSDReadDataSecure (JxfsMSDReadData readData, int securityInfo, byte testResult, byte[] cim86Info)</i>
Description	Constructor of the class.

6 Codes

6.1 Error Codes

Value	Meaning
JXFS_E_MSD_READFAILURE	No read conditions were satisfied (that is, not all tracks specified in <i>tracksToRead</i> parameter have been read or no Watermark was read).
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.
JXFS_E_MSD_NOTSUPPORTED TRACK	At least one track specified in <i>tracksToRead</i> parameter is not supported by the device.
JXFS_E_MSD_NOTRACKS	No tracks specified in <i>tracksToRead</i> parameter.
JXFS_E_MSD_WRITEFAILURE	No write conditions were satisfied.
JXFS_E_MSD_BADDATA	Data is invalid.
JXFS_E_MSD_NOTSUPPORTED CAP	The service does not have secure capability.
JXFS_E_MSD_PARITY	Parity read error.
JXFS_E_MSD_READ_EOF	Only SS,SE,BCC on track.
JXFS_E_MSD_NO_STRIPE	No magnetic stripe or flux on stripe detected (if device has capability to detect this situation).
JXFS_E_MSD_READ_OTHER	Any other type of read error.

Value	Meaning
JXFS_E_CCD_IOERROR	IO error occurred. No ATR data is available.
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.

Value	Meaning
JXFS_E_MOTOR_MEDIAJAMMED	Media is jammed.
JXFS_E_MOTOR_SHUTTERFAIL	Shutter could not be opened.
JXFS_E_MOTOR_NOMEDIA	There is no media to eject.
JXFS_E_MOTOR_BINFULL	Retain bin is full.

6.2 Status Codes

Value	Meaning
JXFS_S_MSD_MEDIA_STATUS	<i>mediaStatus</i> property has changed.

Value	Meaning
JXFS_S_CCD_MEDIA_STATUS	<i>mediaStatus</i> property has changed.

Value	Meaning
JXFS_S_MOTOR_BIN_STATUS	<i>retainBinStatus</i> property has changed.
JXFS_S_MOTOR_BIN_CARDRE TAINED	<i>retainCardCount</i> property has incremented.

Value	Meaning
JXFS_S_MSD_SEC_STATUS	<i>secureModuleStatus</i> property has changed.
JXFS_S_MSD_SEC_READY	Security module ready.
JXFS_S_MSD_SEC_NOTREADY	Security module not ready.
JXFS_S_MSD_SEC_UNKNOWN	State of the security module cannot be determined with the device in its current state.

6.3 Operation Codes

The following codes identify the operation that generated an `OperationCompleteEvent` or `IntermediateEvent`:

Value	Method
JXFS_O_MSD_READDATA	<i>readData</i> , <i>readWMtrack</i>
JXFS_O_MSD_WRITEDATA	<i>writeData</i>

Value	Method
JXFS_O_CCD_CHIPINIT	<i>chipInit</i>
JXFS_O_CCD_CHIPIO	<i>chipIO</i>

Value	Method
JXFS_O_MOTOR_EJECTCARD	<i>ejectCard</i>
JXFS_O_MOTOR_RETAINCARD	<i>retainCard</i>

The following codes identify the reason for an `IntermediateEvent`:

Value	Meaning
JXFS_I_MSD_NO_MEDIA_PRESENT	The read operation request cannot progress because there is no media inserted.
JXFS_I_MSD_MEDIA_INSERTED	The read operation request continues because a media has been inserted.

Value	Meaning
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.

Value	Meaning
JXFS_I_MOTOR_MEDIATAKEN	The media has been taken by the user.

6.4 Constants

Value	Meaning
JXFS_MSD_TYPE_SWIPE = 1	Swipe/pull through magnetic stripe reader/encoder.
JXFS_MSD_TYPE_DIP = 2	Dip magnetic card reader/encoder.
JXFS_MSD_TYPE_MOTOR = 4	Motorized card reader.
JXFS_MSD_SECTYPE_NOTSUPPORTED	No security module available.
JXFS_MSD_SECTYPE_MMBOX	MMBox module available.
JXFS_MSD_SECTYPE_CIM86	CIM86 module available.

Value	Meaning
JXFS_CCD_TYPE_SWIPE = 1	Swipe/pull through chip card device.
JXFS_CCD_TYPE_DIP = 2	Dip chip card device.
JXFS_CCD_TYPE_MOTOR = 4	Motorized chip card device.
JXFS_CCD_TYPE_CONTACTLESS = 8	Contactless chip card device.

Value	Meaning
JXFS_MOTOR_EJECT	At power off /on card is ejected.
JXFS_MOTOR_EJECT_THEN_RETAIN	At power off /on card is ejected, then, after some seconds, it is retained.
JXFS_MOTOR_NOACTION	At power off /on no action is taken.
JXFS_MOTOR_READ_POSITION	At power off /on card is brought to the read/write position.
JXFS_MOTOR_RETAIN	At power off /on card is retained.

Value	Meaning
JXFS_MSD_SEC_NOCHECK	No security check was requested.
JXFS_MSD_SEC_NOTREADY	Security module was not ready.
JXFS_MSD_SEC_SECFAIL	Security module failed reading media security sign.
JXFS_MSD_SEC_SECOK	Successful security check.