CEN

WORKSHOP

AGREEMENT

CWA 16008-13

August 2009

ICS 35.240.40

English version

J/eXtensions for Financial Services (J/XFS) for the Java Platform - Part 13: Scanner Device Class Interface - Programmer's Reference

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties, the constitution of which is indicated in the foreword of this Workshop Agreement.

The formal process followed by the Workshop in the development of this Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN Management Centre can be held accountable for the technical content of this CEN Workshop Agreement or possible conflicts with standards or legislation.

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

This CEN Workshop Agreement is publicly available as a reference document from the CEN Members National Standard Bodies.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and United Kingdom.

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

Management Centre: Avenue Marnix 17, B-1000 Brussels

Ref. No.:CWA 16008-13:2009 E

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 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 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 Secretariat, and at *http://www.cen.eu/cenorm/sectors/sectors/isss/activity/jxfs_membership.asp*. The specification was agreed upon by the J/XFS Workshop Meeting of 2009-05-6/9 in Brussels, and the final version was sent to CEN for publication on 2009-06-12.

The specification is continuously reviewed and commented in the CEN J/XFS Workshop. The information published in this CWA is furnished for informational purposes only. CEN makes no warranty expressed or implied, with respect to this document. Updates of the specification will be available from the CEN J/XFS Workshop public web pages pending their integration in a new version of the CWA (see *http://www.cen.eu/cenorm/sectors/sectors/isss/activity/jxfs_cwas.asp*).

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

Questions and comments can also be submitted to the members of the J/XFS Forum through the J/XFS Forum web-site *http://www.jxfs.net*.

This CWA is composed of the following parts:

- Part 1: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Base Architecture - Programmer's Reference
- Part 2: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Pin Keypad Device Class Interface - Programmer's Reference
- Part 3: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Magnetic Stripe & Chip Card Device Class Interface - Programmer's Reference
- Part 4: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Text Input/Output Device Class Interface - Programmer's Reference
- Part 5: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Cash Dispenser, Recycler and ATM Device Class Interface - Programmer's Reference
- Part 6: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Printer Device Class Interface - Programmer's Reference
- Part 7: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Alarm Device Class Interface - Programmer's Reference
- Part 8: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Sensors and Indicators Unit Device Class Interface - Programmer's Reference
- Part 9: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Depository Device Class Interface - Programmer's Reference
- Part 10: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Check Reader/Scanner Device Class Interface - Programmer's Reference (deprecated in favour of Part 13)
- Part 11: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Camera Device Class Interface - Programmer's Reference
- Part 12: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Vendor Dependant Mode Specification - Programmer's Reference
- Part 13: J/eXtensions for Financial Services (J/XFS) for the Java Platform – Scanner Device Class Interface - Programmer’s Reference (recommended replacement for Part 10)

This CEN Workshop Agreement is publicly available as a reference document from the National Members of CEN : AENOR, AFNOR, ASRO, BDS, BSI, CSNI, CYS, DIN, DS, ELOT, EVS, IBN, IPQ, IST, LVS, LST, MSA, MSZT, NEN, NSAI, ON, PKN, SEE, SIS, SIST, SFS, SN, SNV, SUTN and UNI.

Comments or suggestions from the users of the CEN Workshop Agreement are welcome and should be addressed to the CEN Management Centre.

Important Notice

This CWA part is the recommended replacement by the Workshop for Part 10 of this CWA, “Check Reader/Scanner Device Class Interface”. document. It is recommended that new Device Service implementations for this device functionalities use this document.

1 Scope

This document describes the Scanner device classes. These classes are basic on the J/XFS architecture 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 allows applications and devices to be distributed within a network. So we have the following layers in J/XFS:

- Application
- Device Control and 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 resides in a central repository.

To support Scanner devices the basic Device Control structure is extended with various properties and methods specific to this device type. The extensions are described on the following pages.

2 Overview

2.1 Description

Device Support within the J/XFS API is available for the following device types:

- **Image Scanner**
  General scanner devices consist of components that allow digital images to be captured as a two dimensional array of brightness values for pixels. These device types are becoming more common place in the banking environment where they are used as independent document devices or integrated into compound devices, as for example, in the case of a cheque scanner with digital image capture capabilities.

- **Barcode Scanner**
  General barcode scanner devices consist of components that emit a laser beam, from a visible laser diode, and then convert the reflected light signal into barcode data.

- **Cheque Scanner**
  General cheque scanner devices consist of components that allow information to be read from a cheque using Magnetic Ink Recognition (MICR) and/or Optical Character Recognition (OCR). These device types may also support the features of an image scanner, providing the capability to capture digital images of cheques.

In addition to the functionality listed above, for the different scanner device types, this specification also supports generic functionality for this class of device: auto feed mechanism, pocket archiving, endorsing, encoding and stamping.

The J/XFS image, barcode and cheque scanner controls use an event driven model. Using these controls, applications acquire data in two steps:

- scanner settings are setup and/or queried, then,
- data is acquired.

Extra processing of the media can be performed during the scanning process or as a separate process.

2.2 Guidance Light Control

Any guidance lights associated with a scanner device are controlled implicitely by the device service.

3 Class Diagram

The following class diagram shows the overall layout of the Scanner interfaces and classes provided by J/XFS.

3.1 Classes and Interfaces

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

Class or interface	Name	Description	Extends / Implements
Interface	**com.jxfs.general.IJxfsConst**	Contains constants that are common to all device control categories.	
Interface	**com.jxfs.control.scn.IJxfsScnConst**	Contains constants that are common to all Scanner controls.	Extends: **IJxfsConst**
Interface	**com.jxfs.control.IJxfsBaseControl**	Contains method and property declarations required by all device controls.	
Interface	**com.jxfs.control.scn.IJxfsScnCommonControl**	Contains method and property declarations required by all Scanner device controls.	Extends: **IJxfsBaseControl**
Interface	**com.jxfs.control.scn.IJxfsBarcodeScanner**	Contains method and property declarations required by all device controls which scan barcodes.	
Interface	**com.jxfs.control.scn.IJxfsImageScanner**	Contains method and property declarations required by all device controls which scan images.	
Interface	**com.jxfs.control.scn.IJxfsChequeScanner**	Contains method and property declarations required by all device controls which scan cheques.	

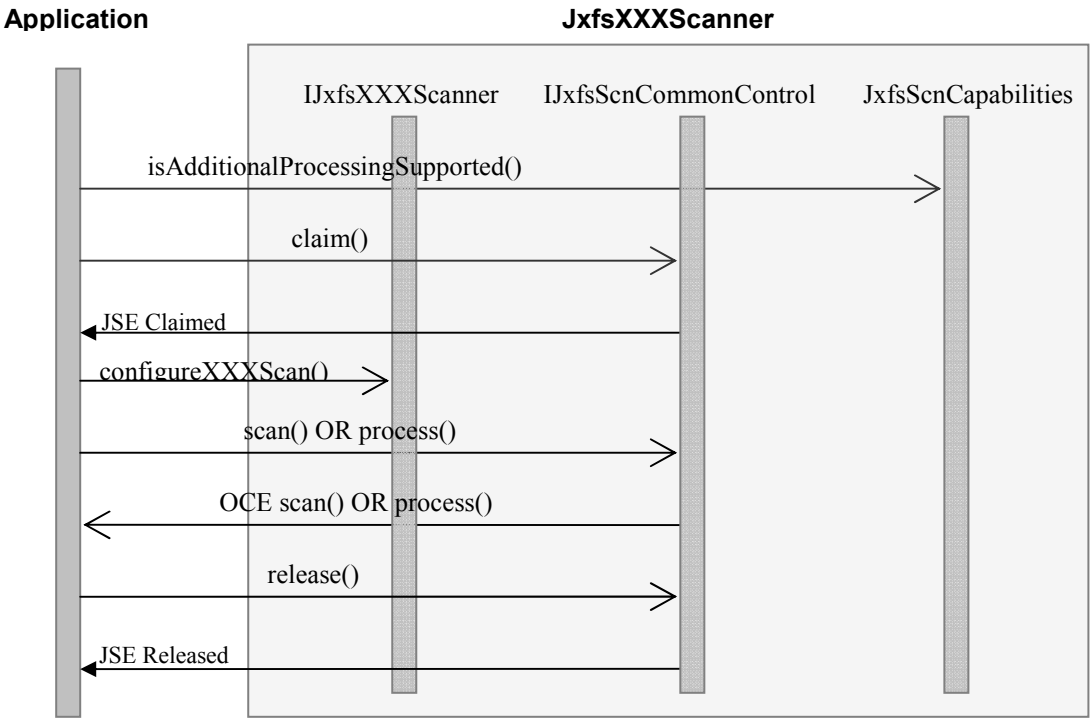
Class	**com.jxfs.control.JxfsBaseControl**	Base class for all device controls.	
Class	**com.jxfs.control.scn.JxfsScnCommonControl**	Abstract Class for the basic Scanner functionality.	Implements: **IJxfsScnCommonControl**
Class	**com.jxfs.control.scn.JxfsChequeScanner**	Base class for all device controls which scan cheques.	Implements: **IJxfsChequeScanner** Extends: **JxfsImageScanner**
Class	**com.jxfs.control.scn.JxfsImageScanner**	Base class for all device controls which scan images.	Implements: **IJxfsImageScanner** Extends: **JxfsScnCommonControl**
Class	**com.jxfs.control.scn.JxfsBarcodeScanner**	Base class for all device controls which scan barcodes.	Implements: **IJxfsBarcodeScanner** Extends: **JxfsScnCommonControl**

4 Device Behaviour

4.1 The Acquiring Process

Shown below are the steps involved in acquiring data for a specific, single media. Some of the steps are optional and/or dependant on device capabilities:

1. Configure acquiring settings: if the attached device is an image or cheque scanner the IJxfsImageScanner.configureImageScan() and IJxfsChequeScanner.configureChequeScan() methods may be used, depending on the capabilities of the device, to configure the device for the next acquiring process. This step is not required if the attached device is a Barcode Scanner.
2. Aquiring data: the IJxfsScnCommonControl.scan() method is executed. Depending on the capabilities of the device, extra processing can be performed during the acquisition stage through the use of the IJxfsScnCommonControl process() and IJxfsScnCommonControl processBundle() methods instead of the IJxfsScnCommonControl.scan() method.
3. Retrieving acquired data: the acquired data is retrieved from the Operation Complete Event returned by the proceeding scan(), process() or processBundle() method call.



The scan(), process() and processBundle() methods are common for all kinds of scanner device, however, the object returned as data in the Operation Complete Event will be different. All possible objects inherit from the JxfsType class.

- For **Barcode Scanners** the data field will contain a **JxfsScnBarcodeResult**.
- For **Image Scanners** the data field will contain a **JxfsScnImageResult**.
- For **Cheque Scanners** the data field will contain a **JxfsScnChequeResult**.

For more information about these objects refer to the Support Classes chapter.

4.2 Extra Processing

Some devices, especially new cheque scanner devices, have the ability to perform extra processing over the media beyond the digital data acquisition process. The process() and processBundle() methods provide support for applying extra processing to the media. An application can determine if a device is able to perform extra processing by querying *IJxfsScnCommonControl.capabilities.additionalProcessingSupported*. Possible extra processing includes:

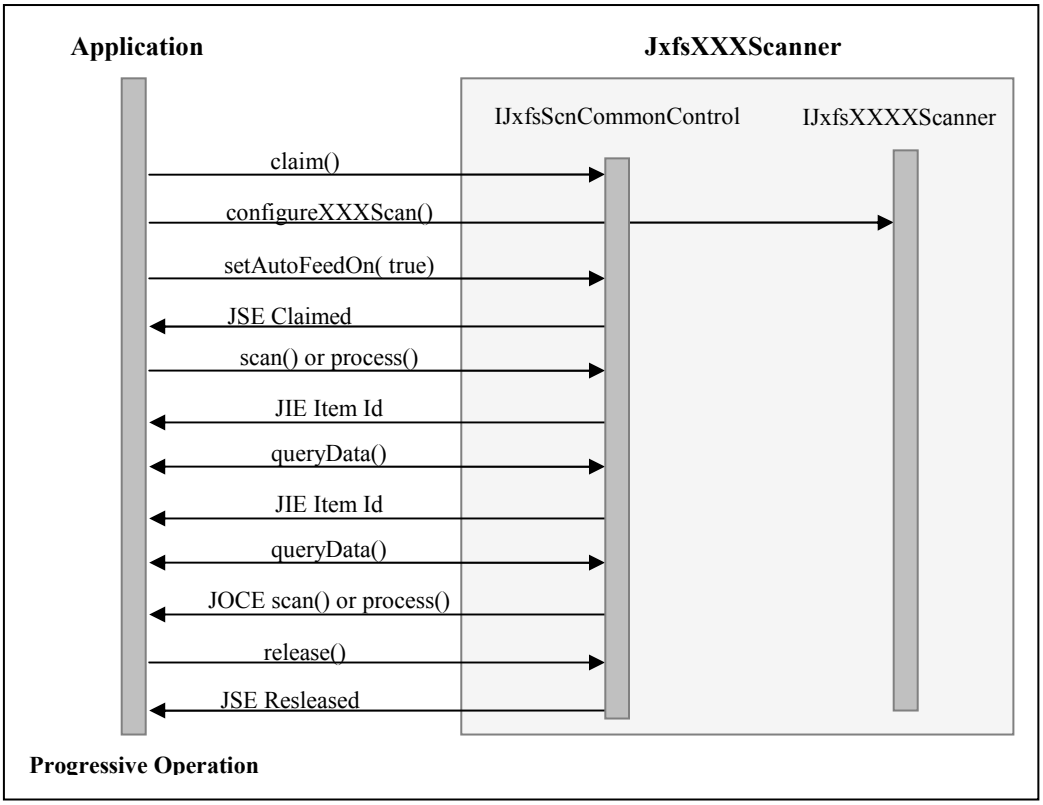
- **Stamping:** a seal can be defined to be stamped on any or both sides of the media. X and Y coordinates for the stamp can also be defined.
- **Encoding:** an alphanumerical String can be encoded on the media.
- **Archiving:** Media can be archived in pockets, whether the processing involves a single item or a bunch of items, a destination pocket or tray, for each individual item it can be specified into which the items will be archived once the processing is complete.
- **Endorsing:** items can be printed to sign, validate or invalidate them using an endorser subdevice.

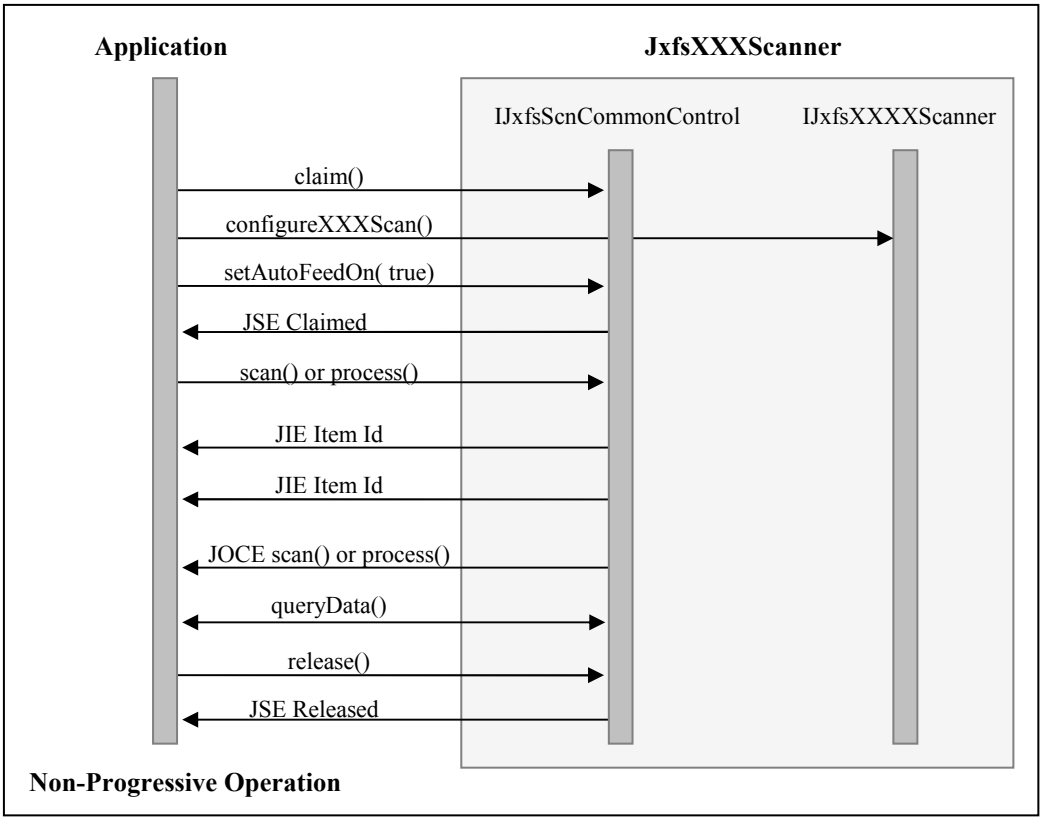
Depending on the specific device, the process() or processBundle() method may implicitely cause image data to be acquired, optionally allow image data to be acquired or not allow image data to be aquire. The specific behaviour is determined from *IJxfsScnCommonControl.capabilities.*scanDuringProcessingSupported. Depending on its value, the JxfsScnProcessData.scanOrder can be used to control whether the process methods acquires image data while performing extra processing.

4.3 AutoFeed Capability

Shown below are the steps involved in acquiring data for a specific media when autofeed capabilities are turned on. Note that this process is common for all device types.

1. Configure acquiring settings: using the configuration methods for the specific device type, the device is configured for the next acquiring process.
2. Enable autofeed: if autofeed processing is optional set IJxfsScnCommonControl.autoFeedOn to *enabled* to ensure device will get media from correct tray/location.
3. Acquiring data: IJxfsScnCommonControl.scan () method is executed depending on the capabilities of the device, extra processing can be performed during the acquisition stage through the use of the process() or processBundle() methods instead of the scan () method.
4. Retrieving acquired data: for each item scanned, the data is stored internally by the device service and assigned an identification number. This identification number is made available through the generation of an Intermediate Event. The *queryData()* method is used to retrieve acquired data. Being synchronous, the method can be called even before the JOCE has been received for progressive retrieval of data.
5. The JOCE of the acquire operation is sent by the device service when no more media is available.


Application
JxfsXXXScanner
IJxfsScnCommonControl
IJxfsXXXXScanner
claim()
configureXXXScan()
setAutoFeedOn(true)
JSE Claimed
scan() or process()
JIE Item Id
queryData()
JIE Item Id
queryData()
JOCE scan() or process()
release()
JSE Resleased
Progressive Operation


Application
JxfsXXXScanner
IJxfsScnCommonControl
IJxfsXXXXScanner
claim()
configureXXXScan()
setAutoFeedOn(true)
JSE Claimed
scan() or process()
JIE Item Id
JIE Item Id
JOCE scan() or process()
queryData()
release()
JSE Released
Non-Progressive Operation

The object returned in the IE of the scan(), process() or processBundle() method will be a JxfsScnDataAvailable object containing an Id for the *queryData()* method.

Note that the scan(), process() or processBundle() method could end with an error after scanning some documents of the whole bunch provided. Even in case of an error the application should get the information related to the IDs returned by the device service while executing the method.

The object returned by the *queryData()* method will be an implementation of *JxfsScnQueryDataResult.* The scan(), process() and processBundle() methods are common for all kind of scanner devices but the object returned as value in the map will be different. All possible objects inherit from JxfsType class.

- For **Barcode Scanners** the data field will contain a **JxfsScnBarcodeResult**.
- For **Image Scanners** the data field will contain a **JxfsScnImageResult**.
- For **Cheque Scanners** the data field will contain a **JxfsScnChequeResult**.

For more information about these objects refer to the Support Classes chapter.

5 Classes and Interfaces

All asynchronous methods return an *identificationID*. If a method cannot be processed because an error was detected before the asynchronous processing of the method began (i.e. before the calling thread returns) a JxfsException is thrown.

After processing has taken place, a *JxfsOperationCompleteEvent* 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 Events, Constants, Error Codes, Exceptions, Status Codes and Support Classes that are used in the methods are described in separate chapters at the end of the documentation.

5.1 Handling of 'null' Input Parameters

A *null* value contained within a parameter class is not allowed, unless explicitly specified. If *null* is passed as a method parameter, a *JxfsException* exception with *errorCode* equal to *JXFS_E_PARAMETER_INVALID* will be thrown.

5.2 Handling of 'null' Return Values

The value null may not be returned as either the result of a method call or contained within a return parameter class unless explicitly specified for the method or parameter class.

5.3 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 read or write a property the application must use the appropriate methods as defined in the JavaBeans specification.

5.3.1 getProperty

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

5.3.2 setProperty

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*: JXFS_E_CLOSED JXFS_E_UNREGISTERED JXFS_E_REMOTE JXFS_E_PARAMETER_INVALID

5.4 IJxfsScnCommonControl

5.4.1 Introduction

The J/XFS General Scanner Device Control interface is defined in *IJxfsScnCommonControl* and *IJxfsBaseControl*. The intent of the J/XFS General Scanner Device Control is to allow data and control to pass between the application and the device support code so that the associated device can be accessed.

This interface inherits all the logic for a generic J/XFS Device Control and also adds functionality to perform scanning and special processing of media. Additional interfaces should be attached to this generic interface in order to provide methods for defining scanning parameters depending on the physical device type. The provided Device Control classes implement this generic scanner device interface plus specific device type interfaces.

5.4.2 Properties

Although *IJxfsScnCommonControl* 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.

5.4.2.1 Summary

Property	Type	Access
capabilities	*JxfsScnCapabilities*	R
autoFeedOn	*JxfsScnAutoFeedOnEnum*	W
scnStatus	*JxfsScnStatus*	R
refusePocketId	*int*	R

Method	Return
get*Property*	*Property*
set*Property*	void
isEndorsable	*boolean*
scan	identificationID
process	identificationID
processBundle	identificationID
queryData	*JxfsScnQueryDataResult*
reset	identificationID
rollback	identificationID
shutterMove	identificationID
retract	identificationID
resetPocketCount	*void*

5.4.2.2 capabilities

Type *JxfsScnCapabilities*
Initial Value Default *JxfsScnCapabilities*.
Description Returns complete information about all common device capabilities.
Events No additional events generated.
Exceptions No additional exceptions thrown.

5.4.2.3 autoFeedOn

Type *JxfsScnAutoFeedOnEnum*
Initial Value unknown until a successful open has completed and the device is in working state.
Description If the device has a configurable autoFeed, as indicated by the *IJxfsScnCommonControl.capabilities.autoFeed* capability, setting this property will enable the automatic document feeder for scanning and autofeed procedures will be accomplished by the device.
This information is persistent through application/device service failure, power failure, open, close and system reset.

The current value can be read using *IJxfsScnCommonControl.scnStatus.autoFeedOn*,

Events Status events will be generated when this property change. Refer to chapter 7.2 for more information.

Exceptions If autoFeed is not configurable (i.e. *IJxfsScnCommonControl.capabilities.autoFeed != configurable*), setting this property to *enabled* will result in a JXFS_E_NOT_SUPPORTED exception being thrown.

5.4.2.4 scnStatus

Type *JxfsScnStatus*

Initial Value Depends on device type.

Description Holds the current status for the different parts of the scanner device. If *JxfsStatus.isOpen()* is *false*, the returned object may not accurately reflect the status of the device.

Events Status events will be generated when members of the *JxfsScnStatus* object change. Refer to chapter 7.2 for more information.

Exceptions No additional exceptions thrown.

5.4.2.5 refusePocketId

Type int

Initial Value JXFS_C_SCN_VALUE_NOT_INITIALIZED until a successful open has completed and the device is in working state.

Description Returns an int representing the id of the pocket for refused items. If the device does not support any pocket, as indicated by the *IJxfsScnCommonControl.capabilities.pocketsSupported* capability, this property stands for JXFS_C_SCN_NOT_SUPPORTED.

If no refuse pocket is indicated (JXFS_C_SCN_UNKNOWN) all refused items will go to the output/reject position selected by the Device Service. Depending on the value of *JxfsScnCapabilities.autoPresent* a further call to *shutterMove* may be needed to put the refused items at customer's disposal.

Events No additional events generated

Exceptions No additional exceptions thrown

5.4.3 Methods

5.4.3.1 isEndorsable

Syntax ***boolean isEndorsable(java.lang.String endorseText) throws JxfsException;***

Description Check if all the chars of the *endorseText* are supported chars in the endorse process

Parameter

Type	Name	Description
java.lang.String	endorseText	Text to be checked if it is supported for endorsing. It can be the full text to be endorsed or parts of it.

Return Value

Type	Meaning
boolean	If *'true'* all the chars of the *endorseText* are supported for the endorse process.

Exceptions This method will throw the JXFS_E_PARAMETER_INVALID exception if *endorseText* is a null reference.

This method must return JXFS_E_NOT_SUPPORTED exception if *IJxfsScnCommonControl.capabilities.endorserCapabilities.endorserSupported* equals 'notSupported' or 'unknown'.

5.4.3.2 scan

Syntax ***identificationID scan(int pocket) throws JxfsException;***

Description This method launches an acquiring process.

If *IJxfsScnCommonControl.capabilities.scanProgressSupported* capability equals *supported*, after the identificationID is returned a series of Intermediate Events will be generated indicating the status of the acquisition process. When the acquisition process completes, a *JxfsOperationCompleteEvent* event is generated to inform the application of the results and to return the acquired data. The class of the returned data object depends on the device control used.

If *IJxfsScnCommonControl.autoFeedOn* property equals *enabled*, the *JxfsOperationCompleteEvent* does not return the acquired data. Instead the application is responsible for retrieving the information. For a detailed description of the retrieval procedure see chapter 4.3

If *JxfsScnCapabilities.escrowSupported* is *notSupported*, after the scanning, the media will be transported to the specified pocket. On the other hand, if *JxfsScnCapabilities.escrowSupported* is *supported* the media will be placed in the escrow (if pocket equals JXFS_C_SCN_ESCROW) or to the specified pocket (if pocket stands for a valid pocket id). If the media can't be placed in the specified pocket, it will be returned to the reject position. If the reject position is not supported or inoperative it will be returned to the output position.

If shutterCmd property in capabilities is *required* then:
- If the input is a tray, the application has to ensure the items are on the tray and the shutter closed before calling scan() J/XFS operation.
- If the input is a slot, the application must open the shutter and call the scan() J/XFS operation right after the shutter opened to start item acceptance.
- When scan completes, the refused items (if any) are not accessible to the customer and the application has to call shutterMove to open/close shutters for the input and refuse positions.

If items remain in the input and refuse position it is the preferred way to clear the input position first and afterwards the refuse position.

Parameter

Type	Name	Description
int	pocket	Specifies the destination pocket, from the available pockets indicated by the *IJxfsScnCommonControl.capabilities.pocketsId* property. If no pockets are supported (*IJxfsScnCommonControl.capabilities.pocketsSupported*) JXFS_C_SCN_VALUE_NOT_INITIALIZED constant must be used.

Events Events, which can be generated by this method.

JxfsOperationCompleteEvent

When the acquiring process is completed, this *JxfsOperationCompleteEvent* is sent to all registered listeners with following data:

Field	Value
operationID	JXFS_O_SCN_SCAN
identificationID	identificationID returned by method.
Result	Common error code or one of the codes listed in *Error Codes.*

data if autoFeed is not being used a *JxfsType* compatible object is returned when the operation completes successfully, otherwise, null is returned.
Refer to *The Acquiring Process* and *AutoFeed Capability* sections for more information on the type of object returned.

JxfsIntermediateEvent

Intermediate Event can be sent by Scanner Device Control to all registered Intermediate Listeners. See section 7.1 for detailed information about Intermediate Events.

JxfsStatusEvent
Status Event can be sent by Scanner Device Control to all registered Status Listeners. See section 7.2 for detailed information about Status Events.

Exceptions This method will throw the JXFS_E_PARAMETER_INVALID exception whether:

- The device support pockets (*JxfsScnCapabilities.pocketsSupported*) and the specified pocket does not exist.
- *If pocket* is JXFS_C_SCN_ESCROW and no escrow is supported (*JxfsScnCapabilities.escrowSupported*).
- If *pocket* is JXFS_C_SCN_VALUE_NOT_INITIALIZED and the device support pockets (*JxfsScnCapabilities.pocketsSupported*).

5.4.3.3 process

Syntax ***identificationID process(JxfsScnProcessData processData, boolean retrieveFromEscrow) throws JxfsException;***

Description This method applies extra processing to the media. It can be used to encode data on the current media, stamp and endorse it, and also move the media to the specified pocket/s if the device supports these capabilities.

Depending on the *IJxfsScnCommonControl.capabilities.scanDuringProcessingSupported* property, this method may or may not acquire image data in addition to performing the specified processing.

- 'notSupported', image data will not be acquired.
- 'alwaysScan',image data will be acquired.
- 'optionalScan', image data may or may not be acquiring depending on the value of the *JxfsScnProcessData.scan* argument.

If *IJxfsScnCommonControl.capabilities.scanProgressSupported* is *supported* and this operation is going to acquire image data, a series of Intermediate Events will be generated indicating the status of the acquiring process. When the acquiring process completes a *JxfsOperationCompleteEvent event* is generated to inform the application of the results and to return the acquired data, if image data was acquired. The class of the returned data object depends on the device control used. If autoFeed is being used, the *JxfsOperationCompleteEvent* does not return the acquired data. Instead the application is responsible for retrieving the information. For a detailed description of the retrieval procedure see chapter 4.3.

If *retrieveFromEscrow* is true and *IJxfsScnCommonControl.capabilities.escrowSupported* is *supported*, the media will be obtained from the escrow. If *retrieveFromEscrow* is *false* the media will be obtained from the input position in any case.

If *IJxfsScnCommonControl.autoFeedOn* is enabled all items will be processed with the same processData. To apply different processData to different items *processBundle()* method must be used.

The *JxfsScnProcessData.itemId* property will be ignored.

If shutterCmd property in capabilities is *required* then:
- If the input is a tray, the application has to ensure the items are on the tray and the shutter closed before calling process() J/XFS operation.
- If the input is a slot, the application must open the shutter and call the process() J/XFS operation right after the shutter opened to start item acceptance.
- When process completes, the refused items (if any) are not accessible to the customer and the application has to call shutterMove to open/close shutters for the input and refuse positions.

If items remain in the input and refuse position it is the preferred way to clear the input position first and afterwards the refuse position.

Parameter

Type	Name	Meaning
JxfsScnProcessData	processData	Object that holds all the required data for media processing.
boolean	retrieveFromEscrow	Specifies if the media is obtained from the escrow (*true*) or from the input position (*false*)

Events

Events, which can be generated by this method.

JxfsOperationCompleteEvent

When the process is completed, this *JxfsOperationCompleteEvent* is sent to all registered listeners with following data:

Field	Value
operationID	JXFS_O_SCN_PROCESS
identificationID	identificationID returned by method.
result	Common error code or one of the codes listed in *Error Codes*.
data	If the operation completed successfully, data was acquired and autofeed was not used, a *JxfsType* compatible object is returned, otherwise, *null* is returned. Refer to *The Acquiring Process* and *AutoFeed Capability* sections for more information on the type of object returned.

Intermediate Event

Intermediate Event can be sent by Scanner Device Control to all registered Intermediate Listeners. See section 7.1 for detailed information about Intermediate Events.

JxfsStatusEvent
Status Event can be sent by Scanner Device Control to all registered Status Listeners. See section 7.2 for detailed information about Status Events.

Exceptions

This method will throw the JXFS_E_PARAMETER_INVALID exception whether:
- *If retrieveFromEscrow* is *true* and no escrow is supported (*JxfsScnCapabilities.escrowSupported*).
- The *JxfsScnProcessData.pocket* specified does not exist or no pockets are supported.
- If *JxfsScnProcessData.encodeData* is not an empty string and *IJxfsScnCommonControl.capabilities.encoderCapabilities.encoderSupported* is *notSupported*.
- If *JxfsScnProcessData.endorseFront/endorseRear* is *true* and *IJxfsScnCommonControl.capabilities.endorserCapabilities.endorserSupported* is *notSupported*.
- If *JxfsScnProcessData.stampFront/stampRear* is *true* and *IJxfsScnCommonControl.capabilities.stampCapabilities.supported* is *notSupported*.

- If stamp is supported and *JxfsScnProcessData.stampFront/RearX* or *stampFront/RearY* are:
  - Negative.
  - Wrong value whether in range (see *JxfsScnStampCapabilities*.maxStampX/Y) or precision.

5.4.3.4 processBundle

Syntax ***identificationID processBundle(java.util.List processDataList, boolean retrieveFromEscrow) throws JxfsException;***

Description This method applies extra processing to the media. It can be used to encode data on the current media, stamp and endorse it, and also move the media to the specified pocket/s if the device supports these capabilities.

Depending on the *IJxfsScnCommonControl.capabilities.scanDuringProcessingSupported* property, this method may or may not acquire image data in addition to performing the specified processing.

- 'notSupported', image data will not be acquired.
- 'alwaysScan',image data will be acquired.
- 'optionalScan', image data may or may not be acquiring depending on the value of the *JxfsScnProcessData.scanOrder* argument.

If *IJxfsScnCommonControl.capabilities.scanProgressSupported* is *supported* and this operation is going to acquire image data, a series of Intermediate Events will be generated indicating the status of the acquiring process. When the acquiring process completes a *JxfsOperationCompleteEvent event* is generated to inform the application of the results and to return the acquired data, if image data was acquired. The class of the returned data object depends on the device control used. For a detailed description of the retrieval procedure see chapter 4.3.

If *retrieveFromEscrow* is *false* the media will be obtained from the input position in any case. It will be Device Service responsibility to stop as soon as possible if *processData* list size don't match item count or if an error occurs.

If *retrieveFromEscrow* is *true* and *IJxfsScnCommonControl.capabilities.escrowSupported* is *supported*, the media will be obtained from the escrow in this case *processData* list size must match the escrow count and all of the *JxfsScnProcessData* list elements must have a valid and unique *itemId* for each of the items present in the escrow.

If shutterCmd property in capabilities is *required* then:

- If the input is a tray, the application has to ensure the items are on the tray and the shutter closed before calling processBundle() J/XFS operation.
- If the input is a slot, the application must open the shutter and call the processsBundle() J/XFS operation right after the shutter opened to start item acceptance.
- When processBundle completes, the refused items (if any) are not accessible to the customer and the application has to call shutterMove to open/close shutters for the input and refuse positions.

If items remain in the input and refuse position it is the preferred way to clear the input position first and afterwards the refuse position.

Parameter

Type	Name	Meaning
java.util.List of JxfsScnProcessData	processData	Object that holds all the required data for media processing.
boolean	retrieveFromEscrow	Specifies if the media is obtained from the escrow (*true*) or from the input position (*false*)

Events Events, which can be generated by this method.

JxfsOperationCompleteEvent

When the process is completed, this *JxfsOperationCompleteEvent* is sent to all registered listeners with following data:

Field	**Value**
operationID	JXFS_O_SCN_PROCESS_BUNDLE
identificationID	identificationID returned by method.
result	Common error code or one of the codes listed in *Error Codes.*
data	None

Intermediate Event

Intermediate Event can be sent by Scanner Device Control to all registered Intermediate Listeners. See section 7.1 for detailed information about Intermediate Events.

JxfsStatusEvent
Status Event can be sent by Scanner Device Control to all registered Status Listeners. See section 7.2 for detailed information about Status Events.

Exceptions This method must return JXFS_E_NOT_SUPPORTED exception whether:

- the device is not able to handle autofeed operations (see *IJxfsScnCommonControl.capabilities.autoFeed* capability in order to know autofeed capabilities in the device).
- *If retrieveFromEscrow* is *true* and JxfsScnCapabilities.escrowSupported is *notSupported*.

This method will throw the JXFS_E_PARAMETER_INVALID exception whether:

- If the *processData* list is: null, empty or list objects with wrong types.
- If *retrieveFromEscrow* is *true* and *processData* size don't match escrow count.
- If *retrieveFromEscrow* is *true* and all items present in the escrow do not have their corresponding element in the *processData* list.
- For all *processData* elements:
  - If *JxfsScnProcessData.pocket* is JXFS_C_SCN_ESCROW and no escrow is supported (*JxfsScnCapabilities.escrowSupported*).
  - If *JxfsScnProcessData*.pocket specified does not exist or no pockets are supported.
  - If *JxfsScnProcessData.encodeData* is not an empty string and *IJxfsScnCommonControl.capabilities.encoderCapabilities.encoderSupported* is *notSupported*.
  - If *JxfsScnProcessData.endorseFront/endorseRear* is *true* and *IJxfsScnCommonControl.capabilities.endorserCapabilities.endorserSupported* is *notSupported*.
  - If *JxfsScnProcessData.stampFront/stampRear* is *true* and *IJxfsScnCommonControl.capabilities.stampCapabilities.supported* is *notSupported*.
  - If stamp is supported and *JxfsScnProcessData.stampFront/RearX* or *stampFront/RearY* are:
    - Negative.
    - Wrong value either in range (see *JxfsScnStampCapabilities*.maxStampX/Y) or precision.

5.4.3.5 queryData

Syntax ***JxfsScnQueryDataResult queryData(int[] dataIds) throws JxfsException;***

Description This method retrieves the scanned media data associated with the supplied identification numbers.

This operation completes successfully if and only if data for all identification numbers provided by the *dataIds* parameter is available.

Scanned media data is stored persistently such that it can be retrieved following an application/device service failure, power failure, open, close and system reset. The data will be available until the next subsequent *scan(), process()* or *processBundle()* job starts executing.

Parameter

Type	Name	Meaning
int[]	dataIds	List of media data identification numbers. The numbers used as Ids are device service specific.

Return Value

Type	Meaning
JxfsScnQueryDataResult	Holds an associative map of data identification numbers and data information objects. Refer to chapter 4.3 for more information.

Exceptions This method must return JXFS_E_NOT_SUPPORTED exception in case the device is not able to handle autofeed operations (see *IJxfsScnCommonControl.capabilities.autoFeed* capability in order to know autofeed capabilities in the device).
This method will throw the JXFS_E_PARAMETER_INVALID exception if:

- the *dataIds* array is null or empty.
- the *dataIds* includes an unknown ID.

5.4.3.6 reset

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

Description This method is used by an application to perform a hardware reset which will attempt to return the device to a known good state. It does not override a lock obtained by another application.

Normally errors are resolved internally by the device service. There are, however, some scenarios in which this automatic recovery may not be performed:

- When automatic recovery will cause an observable impact on the customer. In this case, this method allows the application to decide the best time to perform the recovery.
- When automatic recovery will cause some valuable information to be lost (e.g. information required to deal with a customer dispute).
- When an unrecoverable error has occurred. In this case, the device has to be informed when the error is manually corrected, in order to allow it to perform any device specific activities required to return it to an operational state.

When a reset is required a JXFS_S_SCN_RESET_REQUIRED Status Event will be sent to all registered listeners. Before an application calls this method it should query *JxfsScnStatus.resetStatus* in order to determine the effect of the call.

Events Events, which can be generated by this method.

JxfsOperationCompleteEvent

When a reset operation is completed, this *JxfsOperationCompleteEvent* is sent to all registered listeners with following data:

Field	Value
operationID	JXFS_O_SCN_RESET
identificationID	identificationID returned by method.
result	Common error code or one of the codes listed in *Error Codes*
data	*None*

JxfsIntermediateEvent

Intermediate Event can be sent by Scanner Device Control to all registered Intermediate Listeners. See section 7.1 for detailed information about Intermediate Events.

JxfsStatusEvent

Status Event can be sent by Scanner Device Control to all registered Status Listeners. See section 7.2 for detailed information about Status Events.

5.4.3.7 rollback

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

Description Moves the media or bundle of medias from the internal escrow to the output position if it is supported.

If the device has an escrow then this command is used to rollback the items that are in the escrow to the teller/customer. If there are no items in the escrow an error code JXFS_E_SCN_NOMEDIA is returned on the JxfsOperationCompleteEvent event and the rollback operation is completed.

If shutterCmd property in capabilities is *required*, this command completes when the items has been moved to the rollback position even when it is not accessible to the customer. Then it is the application responsibility to call shutterMove to open/close the shutter.

Subsequent rollback calls are necessary if the device cannot rollback all items in one operation, i.e. if the escrow is larger than the output slot/tray.

If there are more items available to be presented to the user, they can be derived from the result of the rollback operation (a *JxfsScnRollbackResult* object) or by checking the JxfsScnStatus.escrowStatus.contents.count.

Events Events, which can be generated by this method.

JxfsOperationCompleteEvent

When a rollback operation is completed, this *JxfsOperationCompleteEvent* is sent to all registered listeners with following data:

Field	Value
operationID	JXFS_O_SCN_ROLLBACK
identificationID	identificationID returned by method.
result	Common error code or one of the codes listed in *Error Codes*
data	*JxfsScnRollbackResult* object will be returned.

JxfsIntermediateEvent

Intermediate Event can be sent by Scanner Device Control to all registered Intermediate Listeners. See section 7.1 for detailed information about Intermediate Events.

JxfsStatusEvent

Status Event can be sent by Scanner Device Control to all registered Status Listeners. See section 7.2 for detailed information about Status Events.

Exceptions This method must return JXFS_E_NOT_SUPPORTED exception in case the escrow module is not supported (*IJxfsScnCommonControl.capabilities.escrowSupported is notSupported*).

5.4.3.8 shutterMove

Syntax ***identificationID shutterMove(int position, boolean open) throws JxfsException;***

Description This method allows the calling application to open and close a position shutter. The open parameter specifies in which direction the shutter should be moved. The position parameter determines for which position the shutter is moved. If the position is not empty, opening the shutter will move items to a position accessible to the customer, and closing the shutter will move items back if necessary.

This method can only be used if shutterCmd property in capabilities is *required*. Otherwise a JXFS_E_NOT_SUPPORTED completion is returned.

In case shutterCmd property for a given position is *required*, if application is about to perform a shutterMove operation in this position and the physical device decides to implicitely perform the same movement just before, the shutterMove job completes with JXFS_RC_SUCCESSFUL.

Refer to sequence diagrams at the end of the document for usage samples.

Parameter

Type	Name	Meaning
int	position	Shutter to be moved
boolean	open	Indicates if the shutter is going to be opened (*true)* or closed (*false)*

Events Events, which can be generated by this method.

JxfsOperationCompleteEvent

When a *shutterMove* operation is completed, this *JxfsOperationCompleteEvent* is sent to all registered listeners with following data:

Field	Value
operationID	JXFS_O_SCN_SHUTTER_MOVE
identificationID	identificationID returned by method.
result	Common error code or one of the codes listed in *Error Codes*
data	*None*

JxfsIntermediateEvent

Intermediate Event can be sent by Scanner Device Control to all registered Intermediate Listeners. See section 7.1 for detailed information about Intermediate Events.

JxfsStatusEvent

Status Event can be sent by Scanner Device Control to all registered Status Listeners. See section 7.2 for detailed information about Status Events.

Exceptions This method must return JXFS_E_NOT_SUPPORTED exception if:

- *IJxfsScnCommonControl.capabilities.shutterCmd* is *'notRequired'* or *'unknown'*.
- *IJxfsScnCommonControl.capabilities.positionsCapabilities[position] .shutterCmd* is '*false*'

5.4.3.9 retract

Syntax ***identificationID retract(JxfsScnRetractArea retractArea) throws JxfsException;***

Description Retracts the media in the input/output/reject position/s or escrow, to the defined retractArea.

This method may be performed in those devices in which the retract operation is supported, i.e., *JxfsScnCapabilities.retractSupported* is *supported*.

If shutterCmd property in capabilities is *required* then:

- If the position to retract from is a tray, the application has to ensure the items are on the tray and the shutter closed before calling retract() J/XFS operation.
- When retract completes the application has to call shutterMove to open/close shutters for the involved positions.

Events Events, which can be generated by this method.

JxfsOperationCompleteEvent

When a *retract* operation is completed, this *JxfsOperationCompleteEvent* is sent to all registered listeners with following data:

Field	**Value**
operationID	JXFS_O_SCN_RETRACT
identificationID	identificationID returned by method.
result	Common error code or one of the codes listed in *Error Codes*
data	*JxfsScnRetractResult*

JxfsIntermediateEvent

Intermediate Event can be sent by Scanner Device Control to all registered Intermediate Listeners. See section 7.1 for detailed information about Intermediate Events.

JxfsStatusEvent

Status Event can be sent by Scanner Device Control to all registered Status Listeners. See section 7.2 for detailed information about Status Events.

Exceptions This method must return JXFS_E_PARAMETER_INVALID exception in case that a not supported *retractArea* is passed.

5.4.3.10 resetPocketCount

Syntax ***void resetPocketCount (int pocket) throws JxfsException;***

Description Set pocket counter to zero (0).
To be used in physical devices that are not able to keep updated this information, i.e., if *JxfsScnCapabilities.pocketHardwareCountSupported* is *notSupported*.

Parameter

Type	**Name**	**Description**
int	pocket	Specifies the destination pocket, from the available pockets indicated by the *IJxfsScnCommonControl.capabilities.pocketsId* property.

Events Events, which can be generated by this method.

JxfsStatusEvent

Status Event can be sent by Scanner Device Control to all registered Status Listeners. See section 7.2 for detailed information about Status Events.

Exceptions This method will throw the JXFS_E_PARAMETER_INVALID exception whether:
The pocket specifed does not exist or no pockets are supported.

5.5 IJxfsBarcodeScanner

5.5.1 Introduction

This interface provides capabilities to handle a Barcode Scanner. Through this interface it is possible to query barcode device capabilities. The scan process itself is done via the common scanner interface after configuring all parameters.

5.5.2 Properties

Although *IJxfsBarcodeScanner* 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.

5.5.2.1 Summary

Property	Type	Access
barcodeCapabilities	*JxfsScnBarcodeCapabilities*	R

Method	Return
get*Property*	Property

5.5.2.2 barcodeCapabilities

Type *JxfsScnBarcodeCapabilities*
Initial Value Default *JxfsScnBarcodeCapabilities*.
Description Indicates the Bar Code capabilities
Events No additional events generated.
Exceptions No additional exceptions thrown.

5.6 IJxfsImageScanner

5.6.1 Introduction

This interface provides capabilities and methods to handle an image scanner. Through this interface it is possible to query device capabilities and configure the scanning. The scan process itself is done via the common scanner interface after configuring all parameters.

5.6.2 Properties

Although *IJxfsImageScanner* 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.

5.6.2.1 Summary

Property	Type	Access
imageCapabilities	*JxfsScnImageCapabilities*	R

Method	Return
get*Property*	Property
configureImageScan	identificationID

5.6.2.2 imageCapabilities

Type *JxfsScnImageCapabilities*
Initial Value Default *JxfsScnImageCapabilities*
Description Used to keep complete information about all imaging device capabilities.
Events No additional events generated.
Exceptions No additional exceptions thrown.

5.6.3 Methods

5.6.3.1 configureImageScan

Syntax ***identificationID configureImageScan(JxfsScnImageScanParameters parameters) throws JxfsException;***

Description This method configures a subsequent image scan. This method should be called prior to the *scan()*, *process()* or *processBundle()* methods to acquire an image.

Parameter

Type	Name	Meaning
JxfsScnImageScanParameters	parameters	This object holds all parameters that can be defined for an image scanning such as resolution, colour bit depth, etc.

Events Events, which can be generated by this method.

JxfsOperationCompleteEvent

When the *configureImageScan operation* is completed, this *JxfsOperationCompleteEvent* is sent to all registered listeners with following data:

Field	Value
operationID	JXFS_O_SCN_CONFIGURE_IMAGE_SCAN
identificationID	identificationID returned by method.
result	Common error code or one of the codes listed in *Error Codes.*
data	*null* value

Exceptions

This method will throw the JXFS_E_PARAMETER_INVALID exception whether:

- Any of the values specified in parameters.scanMode is not supported by the scanner.
- *parameters.scanMode.brightness* is negative and *IJxfsImageScanner.capabilities.brightnessControl* equals *supported*.
- *parameters.scanMode.gamma* is negative and *IJxfsImageScanner.capabilities.gammaControl* equals *supported*
- *parameters.scanMode.sharpness* is negative and *IJxfsImageScanner.capabilities.sharpnessControl* equals *supported*
- The device cannot scan in the specified area.

5.7 IJxfsChequeScanner

5.7.1 Introduction

This interface provides capabilities and methods to handle cheque scanning functionality. Through this interface it is possible to query device capabilities and configure the scanning. The scan process itself is done via the common scanner interface after configuring all parameters.

5.7.2 Properties

Although *IJxfsChequeScanner* 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.

5.7.2.1 Summary

Property	Type	Access
chequeCapabilities	*JxfsScnChequeCapabilities*	R

Method	Return
get*Property*	Property
configureChequeScan	identificationID

5.7.2.2 chequeCapabilities

Type *JxfsScnChequeCapabilities*
Initial Value Default *JxfsScnChequeCapabilities*.
Description Used to keep complete information about all cheque data acquiring device capabilities.
Events No additional events generated.
Exceptions No additional exceptions thrown.

5.7.3 Methods

5.7.3.1 configureChequeScan

Syntax ***identificationID configureChequeScan(JxfsScnChequeScanParameters parameters) throws JxfsException;***

Description This method configures a subsequent cheque scan. This method should be called prior to the: *scan*(), *process()* or *processBundle()* methods to read a cheque.

Note that if an image of the cheque is also required the *IJxfsImageScanner.configureImageScan()* method should be used to configure the image scanning.

Parameter

Type	Name	Meaning
JxfsScnChequeScanParameters	parameters	This object holds all parameters that can be defined for a cheque data acquire such as image capturing, type of reading, etc.

Events Events, which can be generated by this method.

JxfsOperationCompleteEvent

When the *configureChequeScan* operation is completed, this *JxfsOperationCompleteEvent* is sent to all registered listeners with following data:

Field	**Value**
operationID	JXFS_O_SCN_CONFIGURE_CHEQUE_SCAN
identificationID	identificationID returned by method.
result	Common error code or one of the codes listed in *Error Codes.*
data	null value

Exceptions This method will throw the JXFS_E_PARAMETER_INVALID exception if:

- The device cannot perform the scanning as required.

6 Support Classes

6.1 Summary

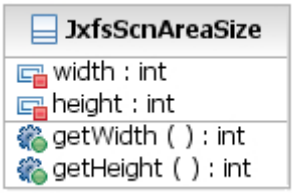
Name	Description
JxfsScnAreaSize	Keeps the size of a media to be scanned.
JxfsScnBarcodeCapabilities	Defines the specific bar code data acquiring capabilities of the scanner device
JxfsScnBarcodeResult	Object returned when a barcode scan operation has completed.
JxfsScnCapabilities	Defines the general capabilities of the scanner device.
JxfsScnChequeCapabilities	Defines the specific cheque data acquiring capabilities of the scanner device
JxfsScnChequeResult	Object returned when a cheque scan operation has completed.
JxfsScnChequeScanParameters	Defines the parameters to acquire data for a cheque.
JxfsScnDataAvailable	Contains the identification of the item notified by the JXFS_I_SCN_DATA_AVAILABLE events.
JxfsScnEncoderCapabilities	Defines the optional encoder module features of the scanner device
JxfsScnEndorserCapabilities	Defines the capabilities of the endorser subdevice, if available.
JxfsScnEndorserData	Defines a line of data to be endorsed by the endorser subdevice.
JxfsScnEscrowContents	Specifies the contents of the escrow.
JxfsScnEscrowStatus	Contains the status of the escrow within the device.
JxfsScnFieldArea	Contains all parameters related to the area of scanning in an acquire process.
JxfsScnImageCapabilities	Defines image capabilities of the scanner device
JxfsScnImageResult	Object returned when an image scan operation has completed.
JxfsScnImageScanParameters	Defines the parameters to acquire data for an image.
JxfsScnMediaCounters	Contains the status of the media.
JxfsScnPocketStatus	Defines a pocket.
JxfsScnPositionCapabilities	Defines the capabilities of a position.
JxfsScnPositionStatus	Contains the status of a position.
JxfsScnPreconfigScanMode	Defines a preconfigured scan mode.
JxfsScnPreconfiguredScanArea	Defines a preconfigured media size available for scanning.
JxfsScnProcessData	Object used to indicate the extra processes that a device service must do over the media.
JxfsScnProcessOperationsResult	Contains the results of all the operations during process.
JxfsScnProgress	Used by Intermediate Event to indicate the progress of an acquire process.
JxfsScnQueryDataResult	Holds the result of the *queryData()* executions.
JxfsScnResetStatus	Provides the information of the consequences of calling the *reset* method.
JxfsScnResult	Abstract basic class of the data returned by the: *JxfsOperationCompleteEvent* event for *scan()*, *process() and processBundle()* operations and the values of the JxfsScnQueryDataResult.dataInformation map returned in the *JxfsOperationCompleteEvent* event for *queryData*()
JxfsScnRetractArea	Provides source and destination details for all items retracted by the device.
JxfsScnRetractResult	Provides details of the *retract* operation result.
JxfsScnResolution	Keeps a valid resolution to acquire an image.
JxfsScnScanMode	Specifies the available scan colour modes and bit depths
JxfsScnShutterStatus	Current status of the shutter module.
JxfsScnStampCapabilities	Defines the capabilities of the stamping subdevice, if available.
JxfsScnStatus	Contains the status of the device.
JxfsScnRollbackResult	Contains the result for *rollback* operations.

6.1.1 JxfsScnAreaSize

6.1.1.1 Usage

This class keeps an image scan size in pixel units. It is used to indicate maximum size for an image and to indicate the size of the area that is going to be scanned through the *JxfsScnFieldArea* class.

6.1.1.2 Class Hierarchy



6.1.1.3 Summary

Extends	Implements
JxfsType	

Property	Type	Default Value	Access
width	*int*	JXFS_C_SCN_VALUE_NOT_INITIALIZED	R
height	*int*	JXFS_C_SCN_VALUE_NOT_INITIALIZED	R

Default Constructor	Parameter
JxfsScnAreaSize	Sets all properties to their default values

Constructor	Parameter
JxfsScnAreaSize	width
	height

Method	Return
get*Property*	*Property*

6.1.1.4 Properties

6.1.1.4.1 width

Type int

Remarks Width of media. The value is indicated using *IJxfsScnCommonControl.capabilities.lengthUnit* and *IJxfsScnCommonControl.capabilities.unitBase* properties.

6.1.1.4.2 height

Type int

Remarks Height of media. The value is indicated using *IJxfsScnCommonControl.capabilities.lengthUnit* and *IJxfsScnCommonControl.capabilities.unitBase* properties.

6.1.1.5 Constructors

6.1.1.5.1 JxfsScnAreaSize

Syntax ***public JxfsScnAreaSize (int width, int height) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *width* or *height* values less than or equal to zero

6.1.1.5.2 JxfsScnAreaSize

Syntax ***public JxfsScnAreaSize ()***

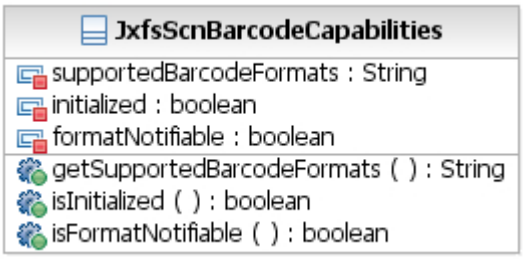
Exceptions No additional exceptions generated.

6.1.2 JxfsScnBarcodeCapabilities

6.1.2.1 Usage

This class defines the cheque data acquiring features of the scanner device. The default object represents the object to be returned, if it is not (yet) known, what the device supports.

6.1.2.2 Class Hierarchy



6.1.2.3 Summary

Extends	Implements
JxfsType	

Property	Type	Default Value	Access
supportedBarcodeFormats	*java.lang.String*	empty string	R
initialized	*boolean*	false	R
formatNotifiable	*boolean*	false	R

Default Constructor	Parameter
JxfsScnBarcodeCapabilities	Sets all properties to its default values

Constructor	Parameter
JxfsScnBarcodeCapabilities	supportedBarcodeFormats
	formatNotifiable

Method	Return
get*Property*	*Property*
is*Property*	*boolean*

6.1.2.4 Properties

6.1.2.4.1 supportedBarcodeFormats

Type *java.lang.String*
Default Value empty string
Remarks Specifies all the device supported bar code formats linked together in a comma separated, single *java.lang.String*.

A *java.lang.String* is used instead of any kind of list in order to make it easily extendable.

An empty string will be allowed for readers that cannot read barcodes or when the status is not yet known

The list of reserved names can be checked in the "*Barcode formats*" section. Proprietary formats can be used depending on device capabilities.

6.1.2.4.2 initialized

Type boolean
Remarks Specifies whether the *supportedBarcodeFormats* and the *formatNotifiable* properties are initialized or not.

6.1.2.4.3 formatNotifiable

Type boolean
Remarks Specifies if the device is able to report the format when reading barcodes.

6.1.2.5 Constructors

6.1.2.5.1 JxfsScnBarcodeCapabilities

Syntax ***public JxfsScnBarcodeCapabilites (java.lang.String supportedBarcodeFormats, boolean formatNotifiable) throws JxfsException***
Description After using this constructor *initialized* property will be always *true*.
Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *supportedBarcodeFormats* is a null reference.

6.1.2.5.2 JxfsScnBarcodeCapabilities

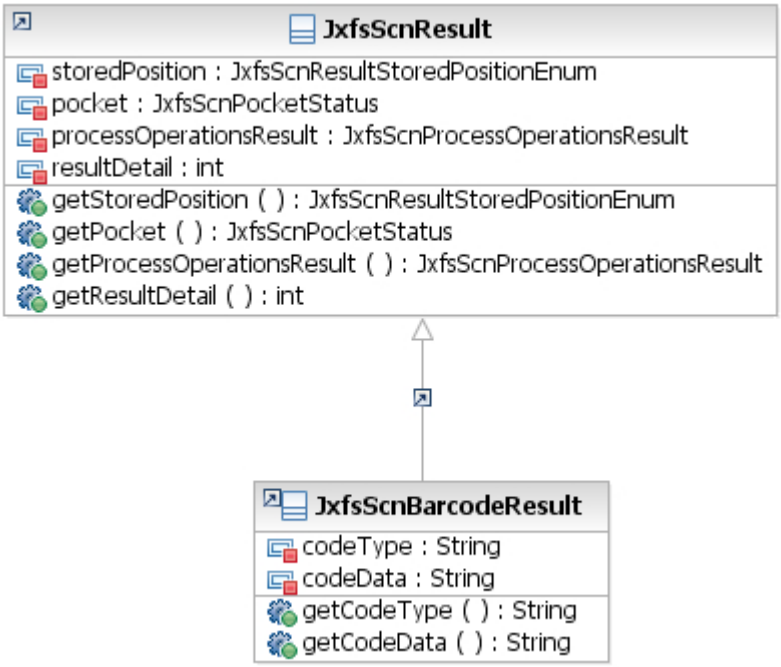
Syntax ***public JxfsScnBarcodeCapabilites ()***
Exceptions No additional exceptions generated.

6.1.3 JxfsScnBarcodeResult

6.1.3.1 Usage

This class contains the data returned by a *JxfsOperationCompleteEvent* event for *scan()*, *process()* and *processBundle()* operations on a barcode scanner device.

6.1.3.2 Class Hierarchy



6.1.3.3 Summary

Extends	Implements
JxfsScnResult	

Property	Type	Access
codeType	*java.lang.String*	R
codeData	*java.lang.String*	R

Constructor	Parameter
JxfsScnBarcodeResult	storedPosition
	pocket
	processOperationsResults
	resultDetail
	codeType
	codeData

Method	Return
get*Property*	*Property*

6.1.3.4 Properties

6.1.3.4.1 storedPosition

Refer to *JxfsScnResult* chapter for information.

6.1.3.4.2 pocket

Refer to *JxfsScnResult* chapter for information.

6.1.3.4.3 processOperationsResults

Refer to *JxfsScnResult* chapter for information.

6.1.3.4.4 resultDetail

Refer to JxfsScnResult chapter for information.

6.1.3.4.5 codeType

Type	java.lang.String
Remarks	Specifies the type of the barcode read. It will be an empty String if *JxfsScnBarcodeCapabilities*.isFormatNotifiable() returns *false*.

6.1.3.4.6 codeData

Type	java.lang.String
Remarks	Contains the read barcode in ASCII format.

6.1.3.5 Constructors

6.1.3.5.1 JxfsScnBarcodeResult

Syntax	***public JxfsScnBarcodeResult (JxfsScnResultStoredPositionEnum storedPosition, JxfsScnPocketStatus pocket, JxfsScnProcessOperationsResult processOperationsResult, int resultDetail, java.lang.String codeType, java.lang.String codeData) throws JxfsException***
Exceptions	Exceptions, which can be generated by this method. JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies: • any of the *JxfsScnResult* constructor. exception cases. • *codeData* is a null reference or an empty String. • *codeType* is a null reference

6.1.4 JxfsScnCapabilities

6.1.4.1 Usage

This class provides information on the common capabilities of the configured device. The default object represents the object to be returned, if it is not (yet) known, what the device supports.

6.1.4.2 Class Hierarchy

6.1.4.3 Summary

Extends	Implements
JxfsType	

Property	Type	Default Value	Access
autoFeed	*JxfsScnAutoFeedKindEnum*	unknown	R
autoSort	*JxfsScnAutoSortEnum*	unknown	R
maxItemsPerBundle	*int*	1	R
encoderCapabilities	*JxfsScnEncoderCapabilities*	Default object	R
endorserCapabilities	*JxfsScnEndorserCapabilities*	Default object	R
additionalProcessingSupported	*JxfsScnAdditionalProcessingSupportEnum*	unknown	R
scanDuringProcessingSupported	*JxfsScnScanDuringProcessingSupportedEnum*	unknown	R
scanProgressSupported	*JxfsScnScanProgressSupportEnum*	unknown	R
lengthUnit	*JxfsScnLengthUnitEnum*	unknown	R
unitBase	*int*	JXFS_C_SCN_UNKNOWN	R
pocketsSupported	*JxfsScnPocketSupportEnum*	unknown	R
stampCapabilities	*JxfsScnStampCapabilities*	Default object	R
escrowSupported	*JxfsScnEscrowSupportEnum*	unknown	R
positionsCapabilities	*JxfsScnPositionCapabilities[]*	Empty array	R
pocketHardwareCountSupported	*JxfsScnPocketHardwareCountSupportEnum*	unknown	R
autoPresent	*JxfsScnAutoPresentEnum*	unknown	R
retractSupported	*JxfsScnRetractSupportedEnum*	unknown	R
supportedRetractAreas	*JxfsScnRetractAreaEnum[]*	Empty array	R
pocketsId	*int[]*	Empty array	R
shutterCmd	*JxfsScnShutterCmdEnum*	unknown	R
defaultInputPosition	*int*	JXFS_C_SCN_UNKNOWN	R
defaultOutputPosition	*int*	JXFS_C_SCN_UNKNOWN	R
defaultRollbackPosition	*int*	JXFS_C_SCN_UNKNOWN	R

Default Constructor	Parameter
JxfsScnCapabilities	Sets all properties to its default values shown in the class hierarchy

Constructor	Parameter
JxfsScnCapabilities	autoFeed
	autoSort
	maxItemsPerBundle
	encoderCapabilities
	endorserCapabilities
	additionalProcessingSupported
	scanDuringProcessingSupported
	scanProgressSupported
	lengthUnit
	unitBase
	pocketsSupported
	stampCapabilities
	escrowSupported
	positionsCapabilities
	pocketHardwareCountSupported
	autoPresent
	retractSupported
	supportedRetractAreas
	pocketsId
	shutterCmd
	defaultInputPosition
	defaultOutputPosition
	defaultRollbackPosition

Method	Return
get*Property*	*Property*

6.1.4.4 Properties

6.1.4.4.1 autoFeed

Type *JxfsScnAutoFeedKindEnum*
Default Value unknown
Remarks Indicates if the device has batch autofeed capability (or Automatic Document Feeder) and if this capability is always activated or is configurable using the *IJxfsScnCommonControl.autoFeedOn* property. If autofeed is configurable and *autoFeedOn* property is *enabled* then whenever a scan or process command is started the complete bunch of media will be accepted into the device sequentially and scanned/processed.

6.1.4.4.2 autoSort

Type *JxfsScnAutoSortEnum*
Default Value unknown
Remarks Indicates if the device can automatically sort media into their corresponding pockets when autofeed processing is used. If the device has no autofeed capabilities then this property has no meaning and will be set to *notSupported*.

6.1.4.4.3 maxItemsPerBundle

Type int
Default Value unknown
Remarks Indicates the maximum number of items which can be scanned in a bundle or stored in the internal escrow.
If *autoFeed* property is *notSupported* then this property equals 1.

6.1.4.4.4 encoderCapabilities

Type *JxfsScnEncoderCapabilities*
Default Value Default object
Remarks Indicates the availability of an encoder module and defines its features.

6.1.4.4.5 endorserCapabilities

Type *JxfsScnEndorserCapabilities*
Default Value Default object
Remarks Indicates the availability of an endorser module and defines its features.

6.1.4.4.6 additionalProcessingSupported

Type *JxfsScnAdditionalProcessingSupportEnum*
Default Value unknown
Remarks Indicates if the device can perform 'extra' processing of the media (e.g. stamping). If *supported*, additional processing of media is possible using the *process()* or *processBundle()* methods. If *notSupported* additional processing of the media is not possible and the *process()* or *processBundle()* methods will return JXFS_E_NOT_SUPPORTED.

6.1.4.4.7 scanDuringProcessingSupported

Type *JxfsScnScanDuringProcessingSupportedEnum*
Default Value unknown
Remarks Indicates whether the *process()* or *processBundle()* methods will acquire image data in addition to performing other media operations. Some devices always perform a scanning when other processes are applied on the inserted media, some others make it optional, and some other cannot perform scans while other tasks are being done. It will be *notSupported* if *additionalProcessSupported* is *notSupported*.

6.1.4.4.8 scanProgressSupported

Type *JxfsScnScanProgressSupportEnum*
Default Value unknown

Remarks Indicates if the device will fire *JXFS_I_SCN_SCAN_PROGRESS* Intermediate Events while the acquiring process is taking place. Some devices can send status information about the scanning process completion while they perform the operation. This can help providing friendlier user interfaces for the final user. If *supported* one or more Intermediate Events will be fired while acquiring data. If *notSupported* no Intermediate Events at all of this kind will be fired.

6.1.4.4.9 lengthUnit

Type *JxfsScnLengthUnitEnum*
Default Value unknown
Remarks Indicates the unit used for length and position properties throughout the device service classes.

6.1.4.4.10 unitBase

Type int
Default Value JXFS_C_SCN_UNKNOWN
Remarks Indicates the resolution as fractions of the lengthUnit value (e.g. a *unitBase* value of 10 and a *lengthUnit* value of MM means that length resolution is 0.1mm).

6.1.4.4.11 pocketsSupported

Type *JxfsScnPocketSupportEnum*
Default Value unknown
Remarks Indicates if device supports handling of one or more pockets where media can be archived after scanning and/or processing. Information about specific pockets is available accesing the *IJxfsScnCommonControl.pockets* property.

6.1.4.4.12 stampCapabilities

Type *JxfsScnStampCapabilities*
Default Value Default object
Remarks Indicates the availability of a stamping module and defines its features.

6.1.4.4.13 escrowSupported

Type *JxfsScnEscrowSupportEnum*
Default Value unknown
Remarks Indicates if device supports internal escrow where media can be archived after/before scanning and/or processing.

6.1.4.4.14 positionsCapabilities

Type *JxfsScnPositionCapabilities[]*
Default Value Empty array.
Remarks Specifies the capabilities of each position supported by the device. An empty array indicates:

- if *pocketsSupported* is *unknown* that this value is unknown
- if *pocketsSupported* is *notSupported*" that no pockets are supported.

Each object in the array reported by this property should contain a unique value for its *position* property, representing a single position. All positions (including default ones) should be part of this array.

6.1.4.4.15 pocketHardwareCountSupported

Type *JxfsScnPocketHardwareCountSupportEnum*
Default Value unknown
Remarks Indicates whether the physical device is able to keep the count of the media stored in pockets or not.
In case of *notSupported*:

- the Device Service will be in charge of counting.
- the *resetPocketCount* method must be used after pocket deplenishment.

6.1.4.4.16 autoPresent

Type *JxfsScnAutoPresentEnum*
Default Value unknown
Remarks Indicates whether the device places the media in an internal position (autoPresent set to *notSupported*) as a result of a *rollback* operation or places the

media at user's disposal to be taken (autoPresent set to *supported*).
In this property is set to *notSupported*, a further call to *shutterMove()* must be performed to put the media available for the user.
If this property is set to *supported*, then the shutterCmd capability will be *notRequired*, as it would not be possible for the calling application to determine when it should open the shutter, due to the possibility for a rollback to be delayed.

6.1.4.4.17 retractSupported

Type *JxfsScnRetractSupportedEnum*
Default Value unknown
Remarks Indicates whether the device is able to retract the media presented to the user or not.

6.1.4.4.18 supportedRetractAreas

Type *JxfsScnRetractAreaEnum[]*
Default Value Empty array until a successful open has completed and the device is in working state.
If *retractSupported* is *notSupported* this property will hold an empty array.
Remarks Specifies the supported retract areas of the device.

6.1.4.4.19 pocketsId

Type int[]
Default Value Empty array until a successful open has completed and the device is in working state.
Remarks Returns an array of int representing the Ids of the supported pockets. If the device does not support any pocket, as indicated by the *IJxfsScnCommonControl.capabilities.pocketsSupported* capability this property will hold an empty array.
The internal escrow will not be included in this list.

6.1.4.4.20 shutterCmd

Type *JxfsScnShutterCmdEnum*
Default Value unknown.
Remarks Defines if explicit shutter handling required. When this property is *required*, the application will be responsible for opening and closing the shutter, using *shutterMove*, for at least one position (see positionsCapabilities for positions).

As a device may have positions with different hardware implementations please refer to JxfsScnPositionCapabilities.shutterCmd for guidance for an individual position.

6.1.4.4.21 defaultInputPosition

Type int
Default Value JXFS_C_SCN_UNKNOWN
Remarks Specifies the default input position to accept medias.
For more information about definition refer to the Position Codes chapter.

6.1.4.4.22 defaultOutputPosition

Type int
Default Value JXFS_C_SCN_UNKNOWN
Remarks Specifies the default output position to place medias.
For more information about definition refer to the Position Codes chapter.

6.1.4.4.23 defaultRollbackPosition

Type int
Default Value JXFS_C_SCN_UNKNOWN
Remarks Specifies the default output position to rollback medias.
For more information about definition refer to the Position Codes chapter.

6.1.4.5 Constructors

6.1.4.5.1 JxfsScnCapabilities

Syntax ***public JxfsScnCapabilities (JxfsScnAutoFeedKindEnum autoFeed, JxfsScnAutoSortEnum autoSort, int maxItemsPerBundle, JxfsScnEncoderCapabilities encoderCapabilites, JxfsScnEndorserCapabilities endorserCapabilities, JxfsScnAdditionalProcessingSupportEnum additionalProcessingSupported, JxfsScnScanDuringProcessingSupportedEnum scanDuringProcessingSupported, JxfsScnScanProgressSupportEnum scanProgressSupported, JxfsScnLengthUnitEnum lengthUnit, int unitBase, JxfsScnPocketSupportEnum pocketsSupported, JxfsScnStampCapabilities stampCapabilities, JxfsScnEscrowSupportEnum escrowSupported, JxfsScnPositionCapabilities[] positionsCapabilities, JxfsScnPocketHardwareCountSupportEnum pocketHardwareCountSupported, JxfsScnAutoPresentEnum autoPresent, JxfsScnRetractSupportedEnum retractSupported, JxfsScnRetractAreaEnum[] supportedRetractAreas, int[] pocketsId, JxfsScnShutterCmdEnum shutterCmd, int defaultInputPosition, int defaultOutputPosition, int defaultRollbackPosition) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies:

- any of the enums or objects is a null reference
- *maxItemsPerBundle* is less than 1.
- *autoFeed* is *notSupported* and *maxItemsPerBundle* not 1.
- *unitBase* is less than 1.
- *supportedRetractAreas* value occurs more than once.
- *shutterCmd* is *required* and none of the *positionsCapabilities* supports shutter.
- any of the default positions (input, output or rollback) is not JXFS_C_SCN_UNKNOWN and is not defined in *positionsCapabilities* array.
- some of the positions defined in *positionsCapabilities* array are not unique.
- *positionsCapabilities* array empty and *pocketsSupported* is *supported*

6.1.4.5.2 JxfsScnCapabilities

Syntax ***public JxfsScnCapabilities ()***

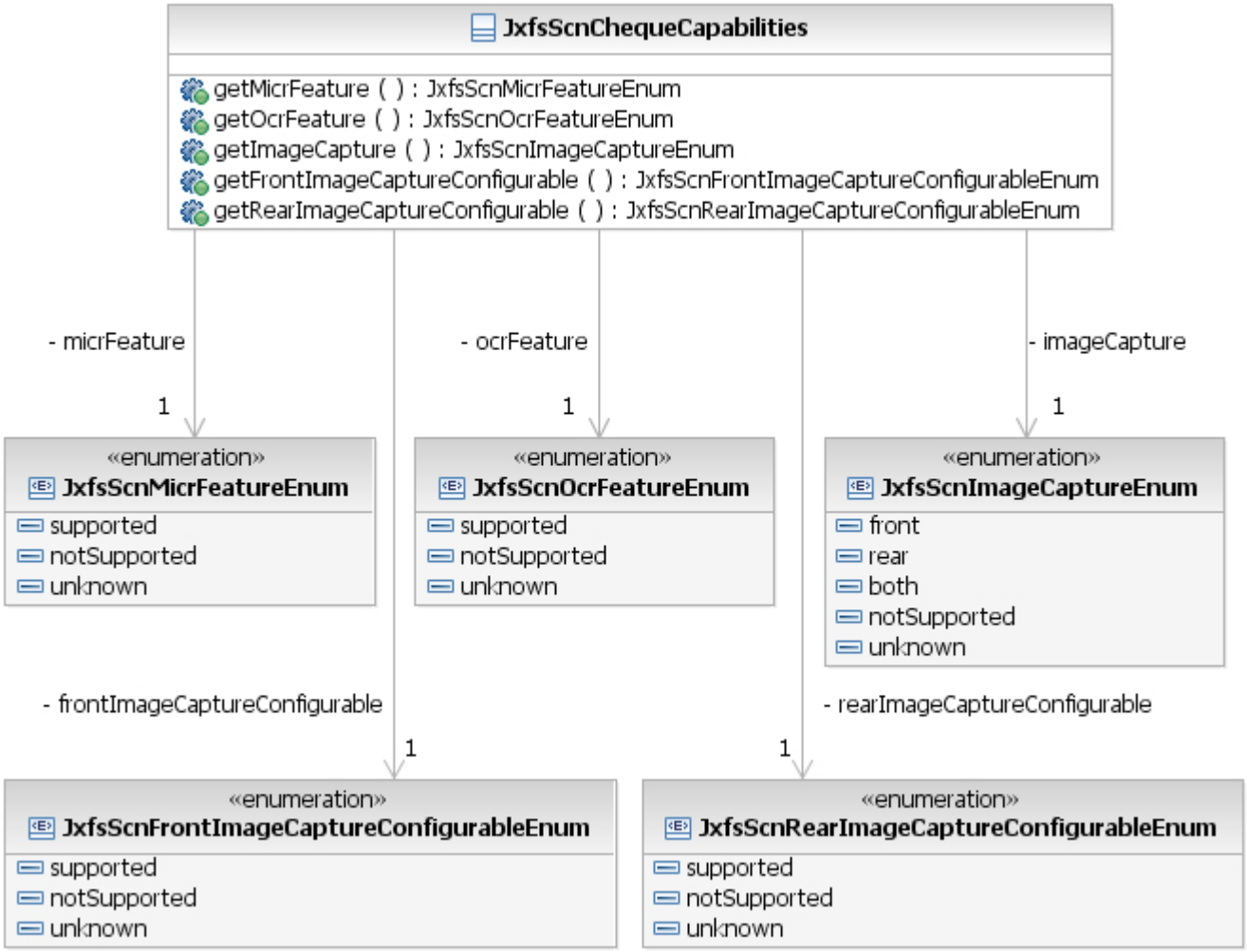
Exceptions No additional exceptions generated.

6.1.5 JxfsScnChequeCapabilities

6.1.5.1 Usage

This class defines the cheque data acquiring features of the scanner device. The default object represents the object to be returned, if it is not (yet) known, what the device supports.

6.1.5.2 Class Hierarchy



6.1.5.3 Summary

Extends	Implements
JxfsType	

Property	Type	Default Value	Access
micrFeature	*JxfsScnMicrFeatureEnum*	unknown	R
ocrFeature	*JxfsScnOcrFeatureEnum*	unknown	R
imageCapture	*JxfsScnImageCaptureEnum*	unknown	R
frontImageCaptureConfigurable	*JxfsScnFrontImageCaptureConfigurable Enum*	unknown	R
rearImageCaptureConfigurable	*JxfsScnRearImageCaptureConfigurable Enum*	unknown	R

Default Constructor	Parameter
JxfsScnChequeCapabilities	Sets all properties to its default values.

Constructor	Parameter
JxfsScnChequeCapabilities	micrFeature
	ocrFeature
	imageCapture
	frontImageCaptureConfigurable
	rearImageCaptureConfigurable

Method	Return
get*Property*	*Property*
is*Property*	*boolean*

6.1.5.4 Properties

6.1.5.4.1 micrFeature

Type *JxfsScnMicrFeatureEnum*
Default Value unknown
Remarks Indicates if the device can read MICR on cheques.

6.1.5.4.2 ocrFeature

Type *JxfsScnOcrFeatureEnum*
Default Value unknown
Remarks Indicates if the device can read OCR on cheques.

6.1.5.4.3 imageCapture

Type *JxfsScnImageCaptureEnum*
Default Value unknown
Remarks Specifies whether the device is capable of acquiring image data, and if so, identifies whether front, rear or both sides of the media can be captured.

6.1.5.4.4 frontImageCaptureConfigurable

Type *JxfsScnFrontImageCaptureConfigurableEnum*
Default Value unknown
Remarks Specifies whether the device can be told to capture the front image or not

6.1.5.4.5 rearImageCaptureConfigurable

Type *JxfsScnRearImageCaptureConfigurableEnum*
Default Value unknown
Remarks Specifies whether the device can be told to capture the rear image or not.

6.1.5.5 Constructors

6.1.5.5.1 JxfsScnChequeCapabilities

Syntax ***public JxfsScnChequeCapabilites (JxfsScnMicrFeatureEnum micrFeature, JxfsScnOcrFeatureEnum ocrFeature, JxfsScnImageCaptureEnum imageCapture, JxfsScnFrontImageCaptureConfigurableEnum frontImageCaptureConfigurable, JxfsScnRearImageCaptureConfigurableEnum rearImageCaptureConfigurable) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies:

- *micrFeature* is a null reference
- *ocrFeature* is a null reference
- *imageCapture* is a null reference
- *frontImageCaptureConfigurableEnum* is a null reference.
- *rearImageCaptureConfigurableEnum* is a null reference.

6.1.5.5.2 JxfsScnChequeCapabilities

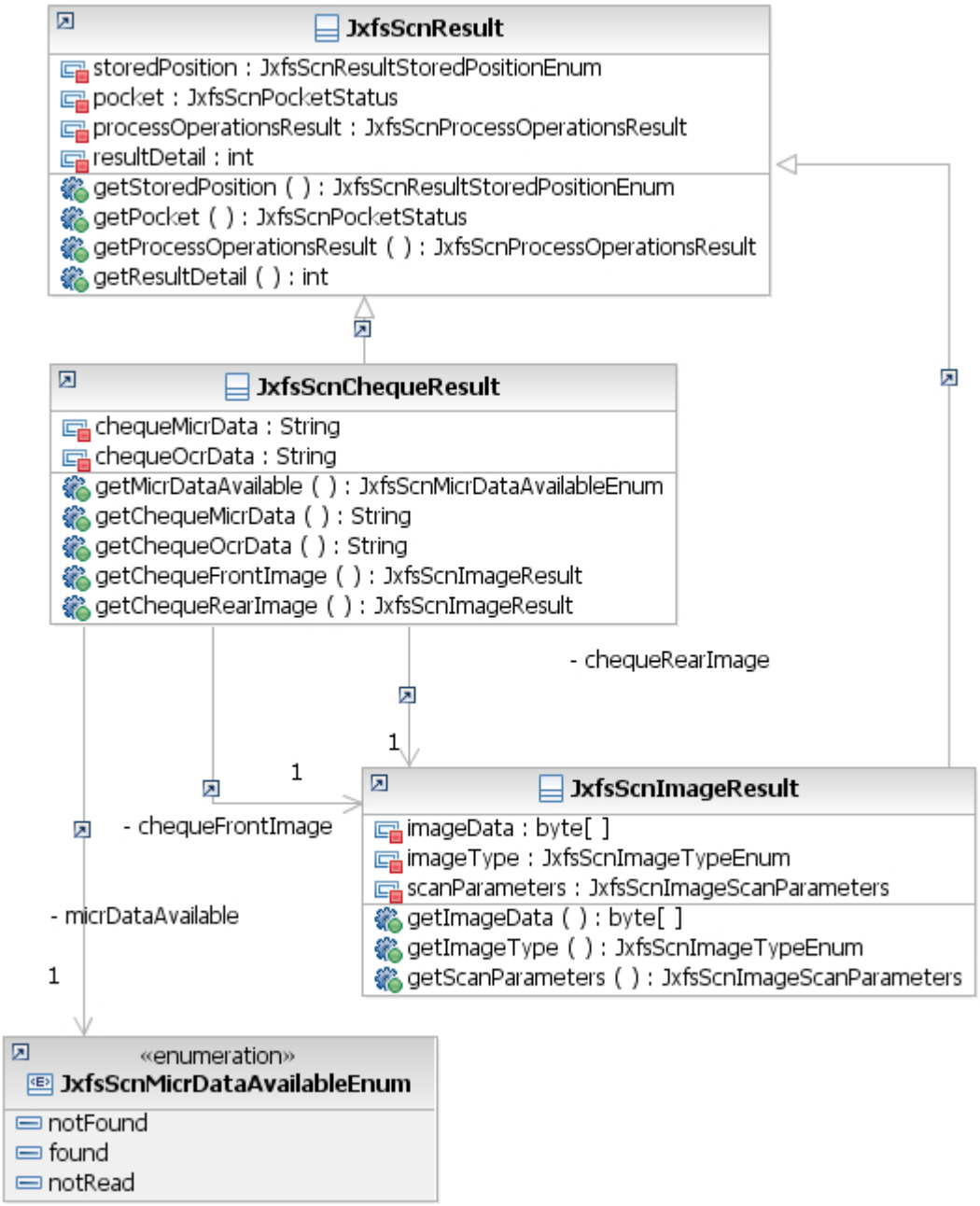
Syntax ***public JxfsScnChequeCapabilites ()***
Exceptions No additional exceptions generated.

6.1.6 JxfsScnChequeResult

6.1.6.1 Usage

This class contains the data returned by a *JxfsOperationCompleteEvent* event for *scan()*, *process()* and *processBundle()* operations when a cheque scanner device is used.

6.1.6.2 Class Hierarchy



6.1.6.3 Summary

Extends	Implements
JxfsScnResult	

Property	Type	Access
micrDataAvailable	*JxfsScnMicrDataAvailableEnum*	R
chequeMicrData	java.lang.String	R
chequeOcrData	java.lang.String	R
chequeFrontImage	*JxfsScnImageResult*	R
chequeRearImage	*JxfsScnImageResult*	R

Constructor	Parameter
JxfsScnChequeResult	storedPosition
	pocket
	processOperationsResults
	resultDetail
	micrDataAvailable
	chequeMicrData
	chequeOcrData
	chequeFrontImage
	chequeRearImage

Constructor	Parameter
JxfsScnChequeResult	storedPosition
	pocket
	processOperationsResults
	resultDetail
	micrDataAvailable
	chequeMicrData
	chequeOcrData

Method	Return
get*Property*	*Property*

6.1.6.4 Properties

6.1.6.4.1 storedPosition

Refer to *JxfsScnResult* chapter for information.

6.1.6.4.2 pocket

Refer to *JxfsScnResult* chapter for information.

6.1.6.4.3 processOperationsResults

Refer to *JxfsScnResult* chapter for information.

6.1.6.4.4 resultDetail

Refer to *JxfsScnResult* chapter for information.

6.1.6.4.5 micrDataAvailable

Type *JxfsScnMicrDataAvailableEnum*
Remarks Specifies if MICR data has been read or not. This property is irrelevant to OCR.

6.1.6.4.6 chequeMicrData

Type java.lang.String
Remarks Contains the MICR data read from the current cheque. If no micr data reading was requested this property will be an empty String.

6.1.6.4.7 chequeOcrData

Type java.lang.String

Remarks Contains the OCR data read from the current cheque. If no ocr data reading was requested this property will be an empty String.

6.1.6.4.8 chequeFrontImage

Type *JxfsScnImageResult*

Remarks Contains the front image data from the current cheque, if requested and available. Otherwise it will return a *JxfsScnImageResult* with "noData" value for *imageType* and an empty array for *imageData*.

6.1.6.4.9 chequeRearImage

Type *JxfsScnImageResult*

Remarks Contains the rear image data from the current cheque, if requested and available. Otherwise it will return a *JxfsScnImageResult* with "noData" value for *imageType* and an empty array for *imageData*.

6.1.6.5 Constructors

6.1.6.5.1 JxfsScnChequeResult

Syntax ***public JxfsScnChequeResult (JxfsScnResultStoredPositionEnum storedPosition, JxfsScnPocketStatus pocket, JxfsScnProcessOperationsResult processOperationsResult, int resultDetail, JxfsScnMicrDataAvailableEnum micrDataAvailable, java.lang.String chequeMicrData, java.lang.String chequeOcrData, JxfsScnImageResult chequeFrontImage, JxfsScnImageResult chequeRearImage) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies:

- any of the *JxfsScnResult* constructor. exception cases.
- *chequeMicrData* is a null reference.
- *chequeOcrData* is a null reference.
- any of the *JxfsScnImageResult* objects is a null reference.
- any of the properties inherited from *JxfsScnResult* except the resultDetail property don't match the same properties of the *JxfsScnImageResult* objects.

6.1.6.5.2 JxfsScnChequeResult

Syntax ***public JxfsScnChequeResult (JxfsScnResultStoredPositionEnum storedPosition, JxfsScnPocketStatus pocket, JxfsScnProcessOperationsResult processOperationsResult, int resultDetail, JxfsScnMicrDataAvailableEnum micrDataAvailable, java.lang.String chequeMicrData, java.lang.String chequeOcrData) throws JxfsException***

Description After using this constructor all *JxfsScnImageResult* objects (*chequeFrontImage* and *chequeRearImage*) will be initialized to:

- "noData" value for *imageType*
- a null reference for scanParameters.
- an empty array for *imageData*.
- the rest of the properties will match the values of *JxfsScnChequeResult* object.

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies:

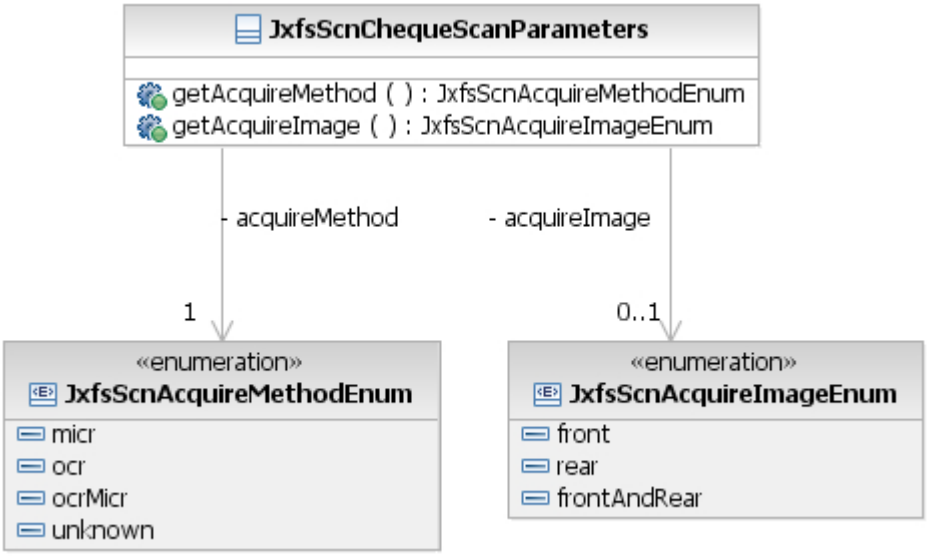
- any of the *JxfsScnResult* constructor. exception cases.
- *chequeMicrData* is a null reference.
- *chequeOcrData* is a null reference.

6.1.7 JxfsScnChequeScanParameters

6.1.7.1 Usage

This class contains the parameters for a cheque acquiring process as requested by the *configureChequeScan()* method.

6.1.7.2 Class Hierarchy



6.1.7.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
acquireMethod	*JxfsScnAcquireMethodEnum*	R
acquireImage	*JxfsScnAcquireImageEnum*	R

Constructor	Parameter
JxfsScnChequeScanParameters	acquireMethod
	acquireImage

Method	Return
get*Property*	*Property*

6.1.7.4 Properties

6.1.7.4.1 acquireMethod

Type *JxfsScnAcquireMethodEnum*

Remarks Indicates how the data should be acquired. Available methods can be queried using *JxfsScnChequeCapabilities.micrFeature* and *JxfsScnChequeCapabilities.ocrFeature* capabilities.

6.1.7.4.2 acquireImage

Type *JxfsScnAcquireImageEnum*

Remarks This property will specify if the scanner shouldn't get an image or which side/s will be scanned. The *JxfsScnChequeCapabilities.imageCapture* property will define if the cheque scanner can perform this operation (and in which side/s) and *JxfsScnChequeCapabilities.frontImageCaptureConfigurable* property will define if the device can be told to scan the front image or not.
For the case that *JxfsScnChequeCapabilities.imageCapture* is notSupported, this property will be *null.*

6.1.7.5 Constructors

6.1.7.5.1 JxfsScnChequeScanParameters

Syntax ***public JxfsScnChequeScanParameters (JxfsScnAcquireMethodEnum acquireMethod, JxfsScnAcquireImageEnum acquireImage) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

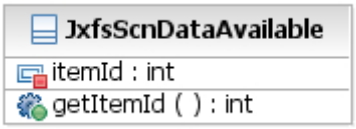
JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *acquireMethod* is a null reference.

6.1.8 JxfsScnDataAvailable

6.1.8.1 Usage

Contains the identification of the item notified by the JXFS_I_SCN_DATA_AVAILABLE events.

6.1.8.2 Class Hierarchy



6.1.8.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
itemId	int	R

Constructor	Parameter
JxfsScnDataAvailable	itemId

Method	Return
get*Property*	*Property*

6.1.8.4 Properties

6.1.8.4.1 itemId

Type int
Remarks Id to be used subsequently by the *queryData* method as a parameter to retrieve the whole data of the scanning.

6.1.8.5 Constructors

6.1.8.5.1 JxfsScnDataAvailable

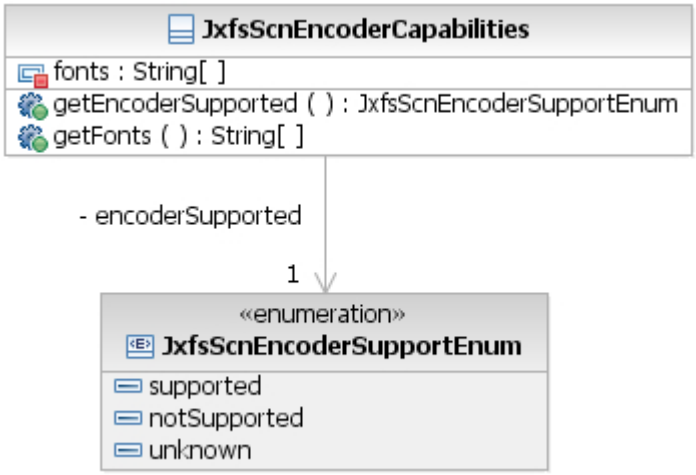
Syntax ***public JxfsScnDataAvailable (int itemId)***
Exceptions No additional exceptions generated.

6.1.9 JxfsScnEncoderCapabilities

6.1.9.1 Usage

This class defines the optional encoder module features of the scanner device. The encoder prints data encoded in whatever encoding fonts the device will support. The default object represents the object to be returned, if it is not (yet) known, what the device supports.

6.1.9.2 Class Hierarchy



6.1.9.3 Summary

Extends	Implements
JxfsType	

Property	Type	Default Values	Access
encoderSupported	*JxfsScnEncoderSupportEnum*	unknown	R
fonts	*java.lang.String[]*	Empty array	R

Default Constructor	Return
JxfsScnEncoderCapabilities	Sets all properties to its default values

Constructor	Parameter
JxfsScnEncoderCapabilities	encoderSupported
	fonts

Method	Return
get*Property*	*Property*

6.1.9.4 Properties

6.1.9.4.1 encoderSupported

Type	*JxfsScnEncoderSupportEnum*
Default Value	unknown
Remarks	Indicates if the device supports an encoder module. Depending on the value of this property, the *JxfsScnProcessData.encodeData* will or will not have any meaning.

6.1.9.4.2 fonts

Type	java.lang.String[]
Default Value	Empty array
Remarks	Indicates the supported fonts for encoding. if the device has an encoder subdevice at least one font must be supported and included in this array. If no encoder subdevice is supported this property holds an empty array. The font indicated in *JxfsScnProcessData.encodeFont* will be one of this array.

6.1.9.5 Constructors

6.1.9.5.1 JxfsScnEncoderCapabilities

Syntax	***public JxfsScnEncoderCapabilites(JxfsScnEncoderSupportEnum encoderSupported, java.lang.String[] fonts) throws JxfsException***
Exceptions	Exceptions, which can be generated by this method. JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies: • *fonts* is a null referece. • *encoderSupported* is a null referece. • *fonts* is an empty array if *encoderSupported* is *supported.*

6.1.9.5.2 JxfsScnEncoderCapabilities

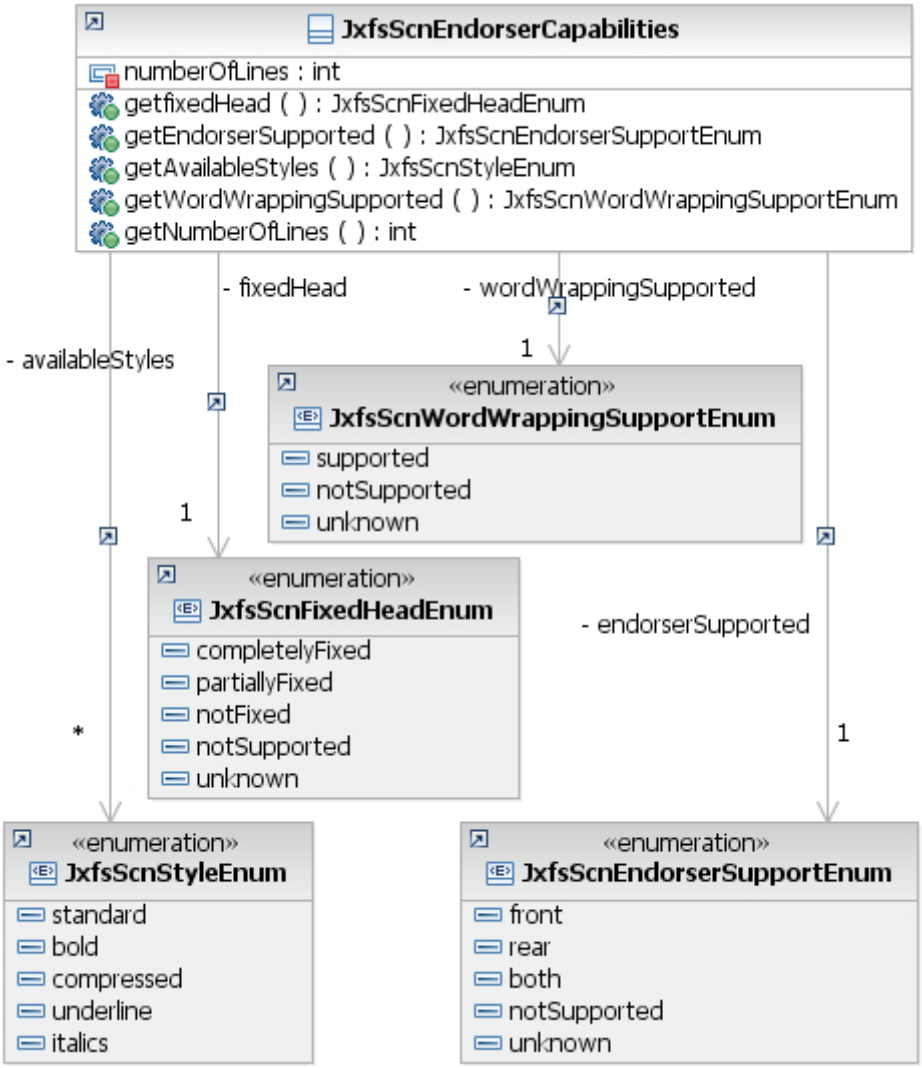
Syntax	***public JxfsScnEncoderCapabilites ()***
Exceptions	No additional exceptions generated.

6.1.10 JxfsScnEndorserCapabilities

6.1.10.1 Usage

This class defines the optional endorser module features of the scanner device. The endorser is a printer-like subdevice within the scanner device. It has the capability to sign / (in)validate the media by printing characters over it while executing the *process()* or *processBundle()* methods. The default object represents the object to be returned, if it is not (yet) known, what the device supports.

6.1.10.2 Class Hierarchy



6.1.10.3 Summary

Extends	Implements
JxfsType	

Property	Type	Default Value	Access
endorserSupported	*JxfsScnEndorserSupportEnum*	unknown	R
availableStyles	*JxfsScnStyleEnum[]*	empty array	R
wordWrappingSupported	*JxfsScnWordWrappingSupportEnum*	unknown	R
fixedHead	*JxfsScnFixedHeadEnum*	unknown	R
numberOfLines	*int*	0	R

Default Constructor	Return
JxfsScnEndorserCapabilities	Sets all properties to its default values

Constructor	Parameter
JxfsScnEndorserCapabilities	endorserSupported
	availableStyles
	wordWrappingSupported
	fixedHead
	numberOfLines

Method	Return
get*Property*	*Property*
is*Property*	*boolean*

6.1.10.4 Properties

6.1.10.4.1 endorserSupported

Type *JxfsScnEndorserSupportEnum*
Default Value unknown
Remarks Indicates if the endorser module exists in the device and what sides of the media can be endorsed.
The properties *JxfsScnProcessData.endorseDataFront* and *endorseDataRear* will have any meaning only if this property is *supported*.

6.1.10.4.2 availableStyles

Type *JxfsScnStyleEnum[]*
Default Value Empty array
Remarks Indicates the available styles for the endorser module. If endorser is not available this property should contain an empty array.
The *JxfsScnProcessData.endorseDataFront[x].style, JxfsScnProcessData.endorseDataRear[x].style* properties will be elements of this array.

6.1.10.4.3 wordWrappingSupported

Type *JxfsScnWordWrappingSupportEnum*
Default Value unknown
Remarks Specifies if the endorser module will perform a word wrapping when endorsing data to the media.

6.1.10.4.4 fixedHead

Type *JxfsScnFixedHeadEnum*
Default Value unknown
Remarks Indicates if the endorser modules head can place text in a user definable position or is fixed in one specific place. If '*completelyFixed'* then head cannot be moved to place text and the *JxfsScnProcessData.endorseDataFront[x].xPosition yPosition and JxfsScnProcessData.endorseDataRear[x].xPosition yPosition* properties won't have any meaning. If '*partiallyFixed'*, at least the Y positions could be specified. If '*notFixed'* then user can define where the text should be placed.
If endorsing is not supported this property will be as well '*notSupported*'

6.1.10.4.5 numberOfLines

Type int
Default Value 0
Remarks Specifies how many lines can be endorsed on the media. Each line can have its own data defined when media is processed. If endorser is not available this property is 0.
The value of this property will define the size of the *java.util.List* of the *JxfsScnProcessData.endorseDataFront* and *endorseDataRear* properties.

6.1.10.5 Constructors

6.1.10.5.1 JxfsScnEndorserCapabilities

Syntax ***public JxfsScnEndorserCapabilites(JxfsScnEndorserSupportEnum endorserSupported, JxfsScnStyleEnum[] availableStyles, JxfsScnWordWrappingSupportEnum wordWrappingSupported, JxfsScnFixedHeadEnum fixedHead, int numberOfLines) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies:

- *endorserSupported* is a null reference.
- *availableStyles* is a null reference.
- *wordWrappingSupported* is a null reference
- *fixedHead* is a null reference.
- *numberOfLines* is less than 0.

6.1.10.5.2 JxfsScnEndorserCapabilities

Syntax ***public JxfsScnEndorserCapabilites ()***

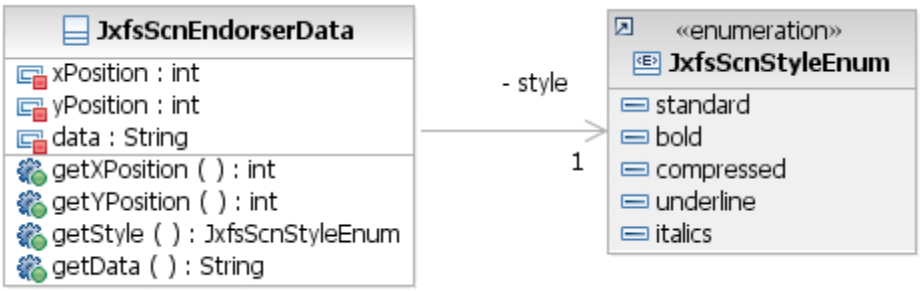
Exceptions No additional exceptions generated.

6.1.11 JxfsScnEndorserData

6.1.11.1 Usage

This class provides properties to specify data to be processed by an endorser module. An object implementing *java.util.List* interface and containing a list of objects of this class can be passed to the *process() or processBundle()* methods through the *JxfsScnProcessData* class.

6.1.11.2 Class Hierarchy



6.1.11.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
xPosition	*int*	R
yPosition	*int*	R
style	*JxfsScnStyleEnum*	R
data	*java.lang.String*	R

Constructor	Parameter
JxfsScnEndorserData	xPosition
	yPosition
	style
	data

Method	Return
get*Property*	*Property*

6.1.11.4 Properties

6.1.11.4.1 xPosition

Type int

Remarks Indicates the X position for endorsing this line on the media If *JxfsScnEndorserCapabilities.fixedHead* is 'completelyFixed' or 'partiallyFixed' this property is not used . The value is indicated using *IJxfsScnCommonControl.capabilities.lengthUnit* and *IJxfsScnCommonControl.capabilities.unitBase* properties.

6.1.11.4.2 yPosition

Type int

Remarks Indicates the Y position for endorsing this line on the media. If *JxfsScnEndorserCapabilities.fixedHead* is 'completelyFixed' this property is not used. The value is indicated using *IJxfsScnCommonControl.capabilities.lengthUnit* and *IJxfsScnCommonControl.capabilities.unitBase* properties.

6.1.11.4.3 style

Type *JxfsScnStyleEnum*

Remarks Indicates the style to be used to endorse data on the media. It must be supported by the endorser module according to the *JxfsScnEndorserCapabilities* object.

6.1.11.4.4 data

Type java.lang.String

Remarks Contains the data to be endorsed on the media.
All the chars that make up this String must be supported to be endorsable.

6.1.11.5 Constructors

6.1.11.5.1 JxfsScnEndorserData

Syntax ***public JxfsScnEndorserCapabilites(int xPosition, int yPosition, JxfsScnStyleEnum style, java.lang.String data) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID Generated if one of the following cases applies:

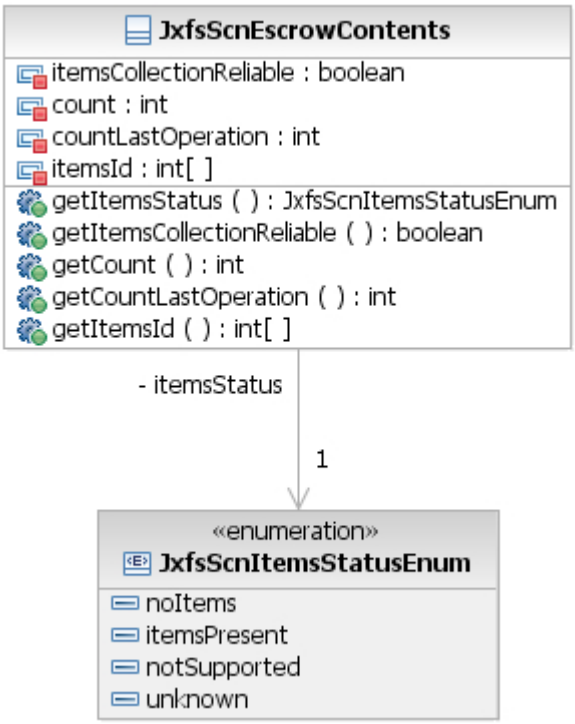
- *xPosition* is less than 0.
- *yPosition* is less than 0.
- *style* is a null reference.
- *data* is a null reference.

6.1.12 JxfsScnEscrowContents

6.1.12.1 Usage

This class contains the result of the internal transport operation.

6.1.12.2 Class Hierarchy



6.1.12.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
itemsStatus	JxfsScnItemsStatusEnum	R
itemsCollectionReliable	*boolean*	R
count	*int*	R
countLastOperation	*int*	R
itemsId	*int[]*	R

Constructor	Parameter
JxfsScnEscrowContents	itemsStatus
	itemsCollectionReliable
	count
	countLastOperation
	itemsId

Method	Return
get*Property*	*Property*
is*Property*	*boolean*

6.1.12.4 Properties

6.1.12.4.1 itemsStatus

Type JxfsScnItemsStatusEnum
Remarks Specifies if there are items present and if so, whether they have been accessible to the customer

6.1.12.4.2 itemsCollectionReliable

Type boolean
Remarks Specifies whether this collection is reliable or not. The reliability of the returned collection is dependent on whether a customer has had access to the items and whether they have been validated following this exposure.

6.1.12.4.3 count

Type int
Remarks Specifies the current count of items within the escrow.

6.1.12.4.4 countLastOperation

Type int
Remarks Specifies the total amount of media in the escrow of the last operation. Normally, the substraction of *JxfsScnMediaCounters*.mediaLastOperation and countLastOperation will give the amount of refused items.

6.1.12.4.5 itemsId

Type int[]
Remarks Specifies the ID of the items present in the escrow.
The size of the array must match *count* value. If there are no items within the escrow this property holds an empty array.

6.1.12.5 Constructors

6.1.12.5.1 JxfsScnEscrowContents

Syntax ***public JxfsScnEscrowContents(JxfsScnItemsStatusEnum itemsStatus, boolean itemsCollectionReliable, int count, int countLastOperation, int[] itemsId) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

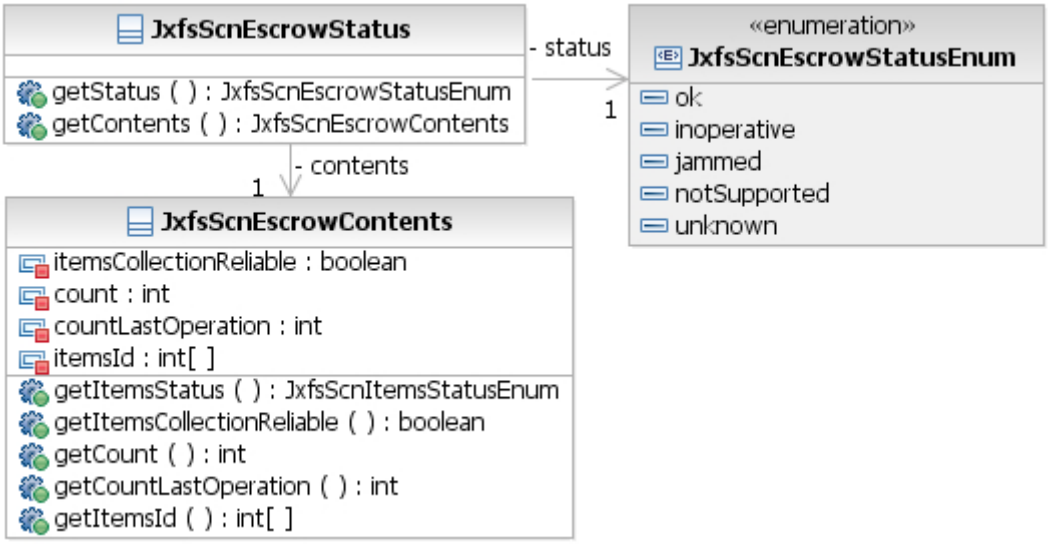
JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *itemsStatus* is a null reference. • *count* is less than 0. • *countLastOperation* is less than 0. • *itemsId* is a null reference. • *itemsId* size don't match count.

6.1.13 JxfsScnEscrowStatus

6.1.13.1 Usage

Specifies whether items are present on the escrow within the device and details whether any items present have been accessible to a customer. An inventory of the available items is also available. Whenever items are removed from or moved to a devices' escrow a JXFS_S_SCN_ESCROW_ITEMS_CHANGED event is generated.

6.1.13.2 Class Hierarchy



6.1.13.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
status	*JxfsScnEscrowStatusEnum*	R
contents	*JxfsScnEscrowContents*	R

Constructor	Parameter
JxfsScnEscrowStatus	status
	contents

Method	Return
get*Property*	*Property*

6.1.13.4 Properties

6.1.13.4.1 status

Type *JxfsScnEscrowStatusEnum*
Remarks Specifies the current status of the escrow module.

6.1.13.4.2 contents

Type *JxfsScnEscrowContents*
Remarks Specifies the result of the internal transport operation

6.1.13.5 Constructors

6.1.13.5.1 JxfsScnEscrowStatus

Syntax ***public JxfsScnEscrowStatus(JxfsScnEscrowStatusEnum status, JxfsScnEscrowContents contents) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

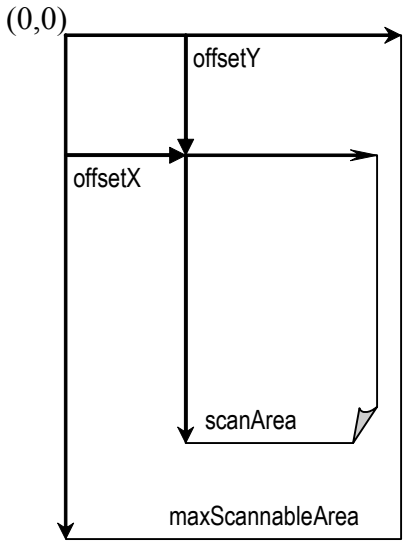
JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies:

- *status* is a null reference.
- *contents* is a null reference.

6.1.14 JxfsScnFieldArea

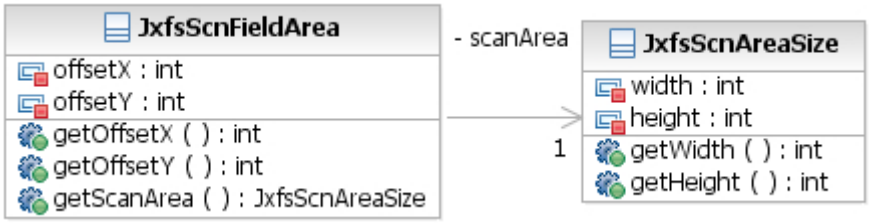
6.1.14.1 Usage

This class identifies an area to be scanned during an image acquisition operation. The maximum scannable area is determined from the *IJxfsScnImageScanner.capabilities.maxScannableArea* property. The following diagram shows how an area to be scanned is specified through the use of this class.



For both the front and the rear sides, the origin of coordinates will be the upper-left position.

6.1.14.2 Class Hierarchy



6.1.14.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
scanArea	*JxfsScnAreaSize*	R
offsetX	*int*	R
offsetY	*int*	R

Constructor	Parameter
JxfsScnFieldArea	scanArea
	offsetX
	offsetY

Method	Return
get*Property*	*Property*

6.1.14.4 Properties

6.1.14.4.1 scanArea

Type *JxfsScnAreaSize*
Remarks Specifies the extents of the area to be scanned.

6.1.14.4.2 offsetX

Type int
Remarks Specifies the horizontal offset of the top-left corner of the area to be scanned. The value is specified using *IJxfsScnCommonControl.capabilities.lengthUnit* and *IJxfsScnCommonControl.capabilities.unitBase* properties. Only allowed values whether in range or in precision will be delivered.

6.1.14.4.3 offsetY

Type int
Remarks Specifies the vertical offset of the top-left corner of the area to be scanned. The value is specified using *IJxfsScnCommonControl.capabilities.lengthUnit* and *IJxfsScnCommonControl.capabilities.unitBase* properties.Only allowed values whether in range or in precision will be delivered.

6.1.14.5 Constructors

6.1.14.5.1 JxfsScnFieldArea

Syntax ***public JxfsScnFieldArea(JxfsScnAreaSize scanArea, int offsetX, int offsetY) throws JxfsException***
Exceptions Exceptions, which can be generated by this method.

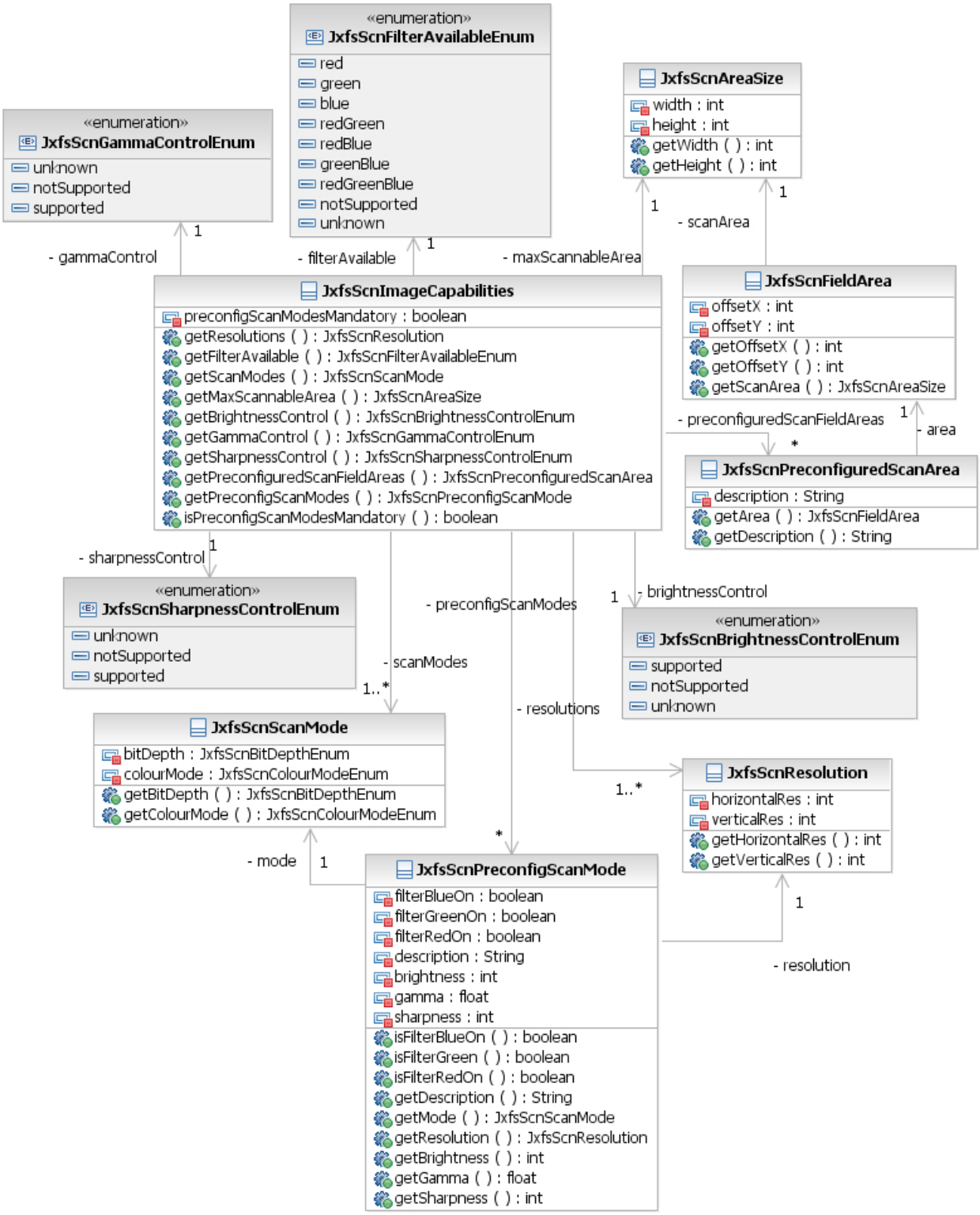
JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *scanArea* is a null reference. • *offsetX* is less than 0. • *offsetY* is less than 0.

6.1.15 JxfsScnImageCapabilities

6.1.15.1 Usage

This class provides information on the image capabilities of the configured device. The default object represents the object to be returned, if it is not (yet) known, what the device supports.

6.1.15.2 Class Hierarchy



6.1.15.3 Summary

Extends	Implements
JxfsType	

Property	Type	Default Value	Access
resolutions	*JxfsScnResolution[]*	Empty array	R
filterAvailable	*JxfsScnFilterAvailableEnum*	unknown	R
scanModes	*JxfsScnScanMode[]*	Empty array	R
maxScannableArea	*JxfsScnAreaSize*	Default object	R
brightnessControl	*JxfsScnBrightnessControlEnum*	unknown	R
gammaControl	*JxfsScnGammaControlEnum*	unknown	R
sharpnessControl	*JxfsScnSharpnessControlEnum*	unknown	R
preconfiguredScanFieldAreas	*JxfsScnPreconfiguredScanArea[]*	Empty array	R
preconfigScanModes	*JxfsScnPreconfigScanMode[]*	Empty array	R
preconfigScanModesMandatory	*boolean*	false	R

Default Constructor	Parameter
JxfsScnImageCapabilities	Sets all properties to their default values

Constructor	Parameter
JxfsScnImageCapabilities	resolutions
	filterAvailable
	scanModes
	maxScannableArea
	brightnessControl
	gammaControl
	sharpnessControl
	preconfiguredScanFieldAreas
	preconfigScanModes
	preconfigScanModesMandatory

Method	Return
get*Property*	*Property*
is*Property*	*boolean*

6.1.15.4 Properties

6.1.15.4.1 resolutions

Type *JxfsScnResolution[]*
Default Value Empty array
Remarks Specifies the allowed resolutions for this scanner to acquire images. Contains a group of *JxfsScnResolution* objects with valid resolutions. At least one valid resolution must be provided.
An empty array will mean that the allowed resolutions are yet unkown.

6.1.15.4.2 filterAvailable

Type *JxfsScnFilterAvailableEnum*
Default Value unknown
Remarks Specifies if hardware can apply a red/green/blue filter over the scanning image.

6.1.15.4.3 scanModes

Type *JxfsScnScanMode[]*
Default Value Empty array
Remarks Specifies the allowed colour modes and bit depths for this scanner to acquire images. Contains a group of *JxfsScnScanMode* objects with valid modes and bit depths. At least one valid scan mode must be provided.
An empty array will mean that the allowed colour modes are yet unkown.

6.1.15.4.4 maxScannableArea

Type	*JxfsScnAreaSize*
Default Value	default object
Remarks	Specifies the maximum media size that can be scanned.

6.1.15.4.5 brightnessControl

Type	*JxfsScnBrightnessControlEnum*
Default Value	unknown
Remarks	Specifies if brightness can be controlled by the application.

6.1.15.4.6 gammaControl

Type	*JxfsScnGammaControlEnum*
Default Value	unknown
Remarks	Specifies if gamma can be controlled by the application. The term gamma refers to the ratio between the input and output light intensity.

6.1.15.4.7 sharpnessControl

Type	*JxfsScnSharpnessControlEnum*
Default Value	unknown
Remarks	Specifies if sharpness can be controlled by the application. The term sharpness refers to the capability to emphasize or de-emphasize the edges of an image.

6.1.15.4.8 preconfiguredScanFieldAreas

Type	*JxfsScnPreconfiguredScanArea[]*
Default Value	Empty array
Remarks	Specifies a list of preconfigured scan field areas that can be handled by the scanner. This can be useful to the application developer because most scanners can only handle a clearly defined set of media sizes. If no preconfigured scan areas are provided by the device service the property holds an empty array.

6.1.15.4.9 preconfigScanModes

Type	*JxfsScnPreconfigScanMode[]*
Default Value	Empty array
Remarks	Specifies a list of preconfigured scan modes that can be used by the scanner. This can be useful to the application developer because most scanners can only handle a clearly defined set of scan modes. If no preconfigured scan modes are provided by the device service the property holds an empty array.

6.1.15.4.10 preconfigScanModesMandatory

Type	*boolean*
Default Value	false
Remarks	Specifies if the list of preconfigured scan modes is mandatory or just a recommendation. If *true* the device supports only the scan modes included on the *preconfigScanModes* list.

6.1.15.5 Constructors

6.1.15.5.1 JxfsScnImageCapabilities

Syntax	***public JxfsScnImageCapabilites()***
Exceptions	No additional exceptions generated.

6.1.15.5.2 JxfsScnImageCapabilities

Syntax	***public JxfsScnImageCapabilites(JxfsScnResolution[] resolutions, JxfsScnFilterAvailableEnum filterAvailable, JxfsScnScanMode[] scanModes, JxfsScnAreaSize maxScannableArea, JxfsScnBrightnessControlEnum brightnessControl, JxfsScnGammaControlEnum gammaControl, JxfsScnSharpnessControlEnum sharpnessControl, JxfsScnPreconfiguredScanArea [] preconfiguredScanFieldAreas, JxfsScnPreconfigScanMode[] preconfigScanModes, boolean preconfigScanModesMandatory) throws JxfsException***
Exceptions	Exceptions, which can be generated by this method.

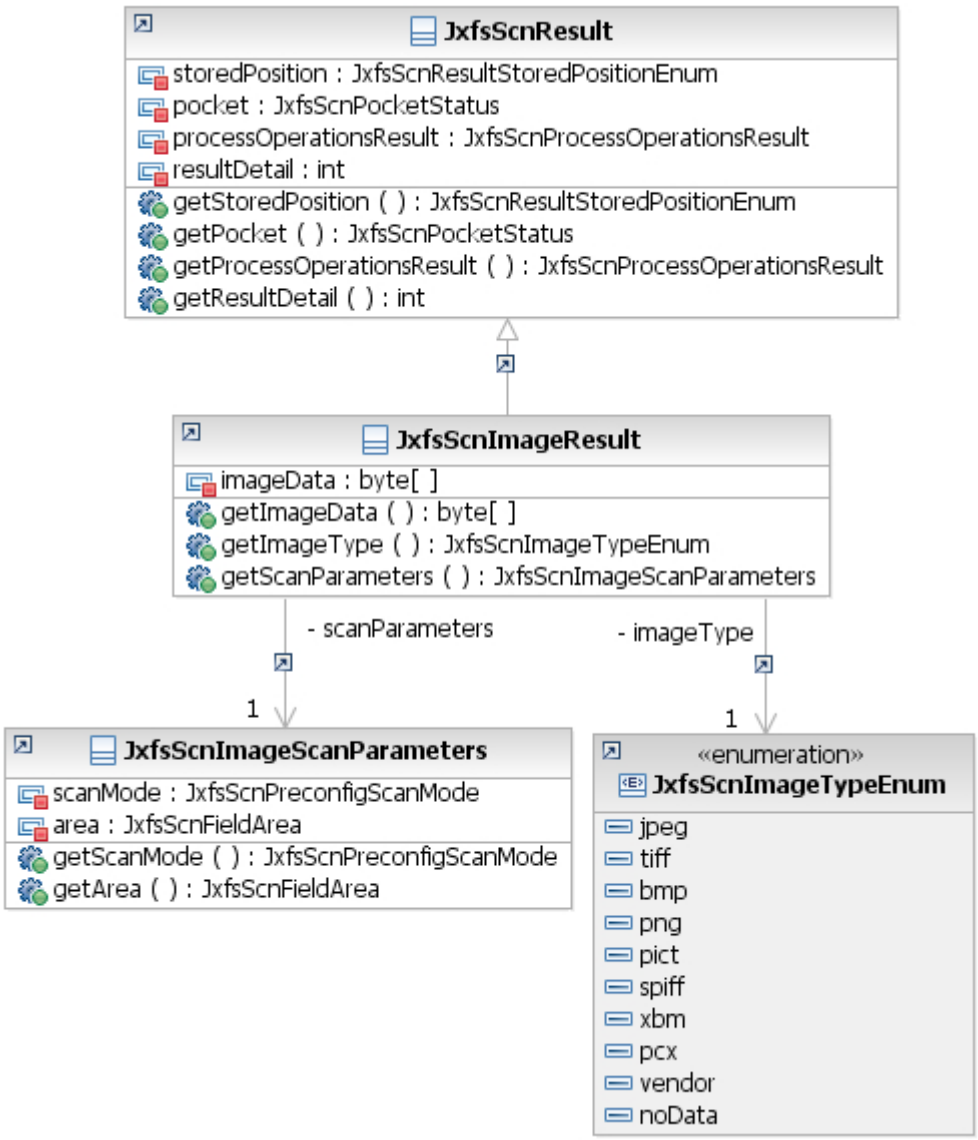
JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *resolutions* is a null reference. • *filterAvailable* is a null reference. • *scanModes* is a null reference. • *maxScannableArea* is a null reference. • *brightnessControl* is a null reference. • *gammaControl* is a null reference. • *sharpnessControl* is a null reference. • *preconfiguredScanFieldAreas* is a null reference. • *preconfigScanModes* is a null reference. • *preconfigScanModesMandatory* is *true* and *preconfigScanModes* is an empty array.

6.1.16 JxfsScnImageResult

6.1.16.1 Usage

This class contains the data returned by an *JxfsOperationCompleteEvent* event for *scan()*, *process()* and *processBundle()* operations for an image scanning.

6.1.16.2 Class Hierarchy



6.1.16.3 Summary

Extends	Implements
JxfsScnResult	

Property	Type	Access
imageData	*byte[]*	R
imageType	*JxfsScnImageTypeEnum*	R
scanParameters	*JxfsScnImageScanParameters*	R

Constructor	Parameter
JxfsScnImageResult	storedPosition
	pocket
	processOperationsResults
	resultDetail
	imageData
	imageType
	scanParameters

Method	Return
get*Property*	*Property*

6.1.16.4 Properties

6.1.16.4.1 storedPosition

Refer to *JxfsScnResult* chapter for information.

6.1.16.4.2 pocket

Refer to *JxfsScnResult* chapter for information.

6.1.16.4.3 processOperationsResults

Refer to *JxfsScnResult* chapter for information.

6.1.16.4.4 resultDetail

Refer to *JxfsScnResult* chapter for information.

6.1.16.4.5 imageData

Type byte[]

Remarks Contains the image data as an array of bytes. The format of the image is vendor dependant and is specified in the *imageType* property. This property is ignored if the resultDetail for this object instance is indicating any kind of error.

6.1.16.4.6 imageType

Type *JxfsScnImageTypeEnum*

Remarks Indicates the format of the data returned by the *imageData* property. This property is ignored if the resultDetail for this object instance is indicating any kind of error.

6.1.16.4.7 scanParameters

Type *JxfsScnImageScanParameters*

Remarks Indicates the parameters used to perform the scan.

6.1.16.5 Constructors

6.1.16.5.1 JxfsScnImageResult

Syntax ***public JxfsScnImageResult(JxfsScnResultStoredPositionEnum storedPosition, JxfsScnPocketStatus pocket, JxfsScnProcessOperationsResult processOperationsResult, int resultDetail, byte[] imageData, JxfsScnImageTypeEnum imageType, JxfsScnImageScanParameters scanParameters) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID Generated if one of the following cases applies:

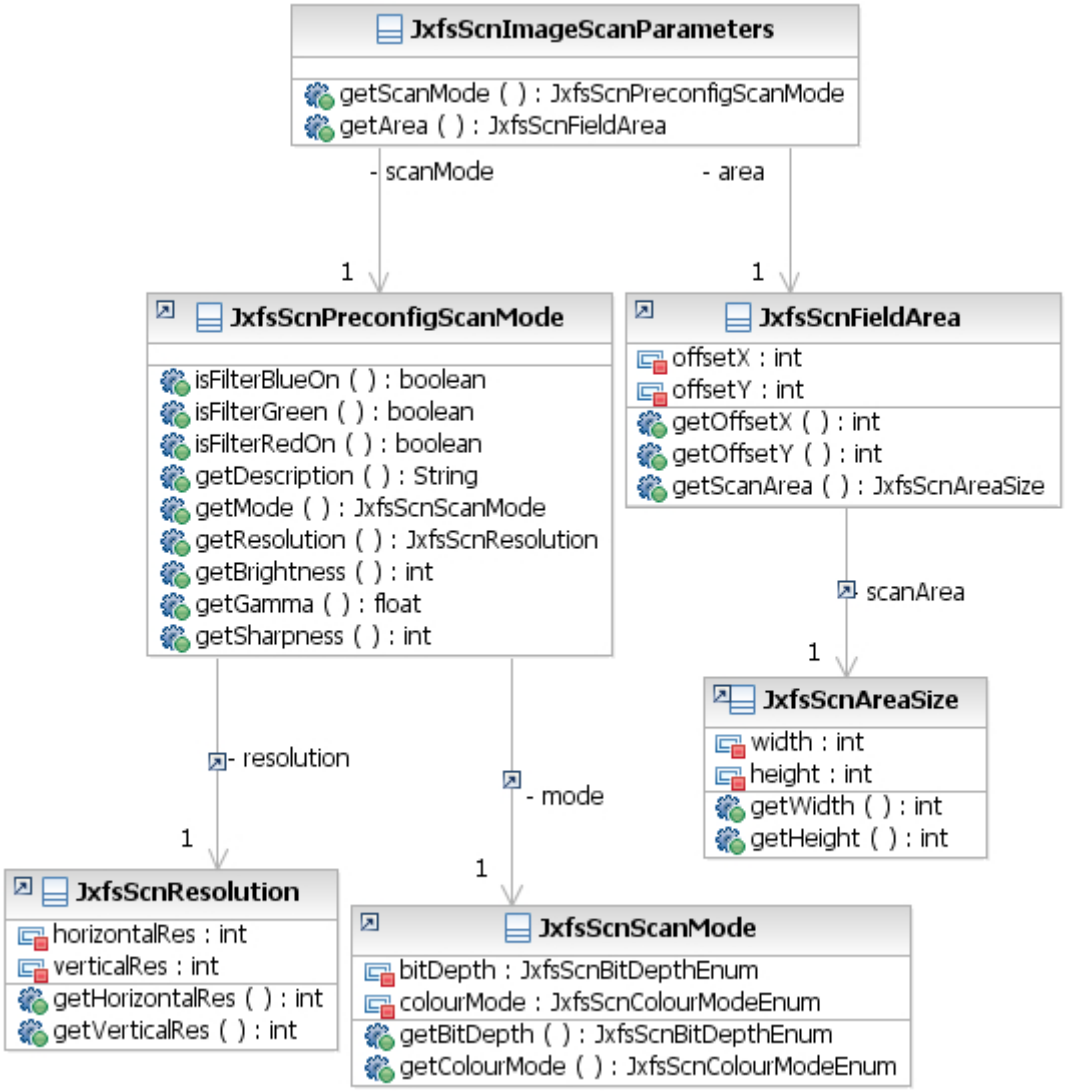
- any of the JxfsScnResult constructor exception cases.
- *imageData* is a null reference.
- *imageData* is an empty array unless *imageType* has 'noData' value.
- *imageType* is a null reference.
- *scanParameters* is a null reference, unless *imageType* has 'noData' value.

6.1.17 JxfsScnImageScanParameters

6.1.17.1 Usage

This class holds a list of parameters needed for an image scanning process. It is used in the *IJxfsImageScanner.configureImageScan()* method.

6.1.17.2 Class Hierarchy



6.1.17.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
scanMode	*JxfsScnPreconfigScanMode*	R
area	*JxfsScnFieldArea*	R

Constructor	Parameter
JxfsScnImageScanParameters	scanMode
	area

Method	Return
get*Property*	*Property*

6.1.17.4 Properties

6.1.17.4.1 scanMode

Type *JxfsScnPreconfigScanMode*

Remarks Preconfigured scan mode that can be used by the scanner. This can be useful to the application developer because most scanners can only handle a clearly defined set of scan modes.

6.1.17.4.2 area

Type *JxfsScnFieldArea*

Remarks Contains area to be scanned in acquire process.

6.1.17.5 Constructors

6.1.17.5.1 JxfsScnImageScanParameters

Syntax ***public JxfsScnImageScanParameters(JxfsScnPreconfigScanMode scanMode, JxfsScnFieldArea area) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID Generated if one of the following cases applies:

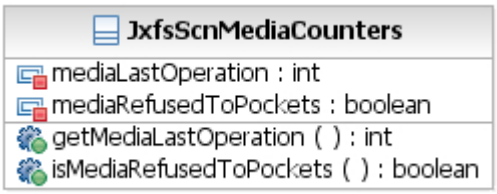
- *scanMode* is a null reference.
- *area* is a null reference.

6.1.18 JxfsScnMediaCounters

6.1.18.1 Usage

This class provides the status information about the media.

6.1.18.2 Class Hierarchy



6.1.18.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
mediaLastOperation	*int*	R
mediaRefusedToPockets	*boolean*	R

Constructor	Parameter
JxfsScnMediaCounters	mediaLastOperation
	mediaRefusedToPockets

Method	Return
get*Property*	*Property*
is*Property*	*boolean*

6.1.18.4 Properties

6.1.18.4.1 mediaLastOperation

Type int

Remarks Indicates the total amount of media issued in the last scan or process operation.

6.1.18.4.2 mediaRefusedToPockets

Type boolean

Remarks Indicates that medias were refused directly to the pockets.
true means that the media were refused directly to the pocket. For this case the amount of media refused can be known by *JxfsScnEscrowContents.countLastOperation*

6.1.18.5 Constructors

6.1.18.5.1 JxfsScnMediaCounters

Syntax ***public JxfsScnMediaCounters (int mediaLastOperation, boolean mediaRefusedToPockets) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

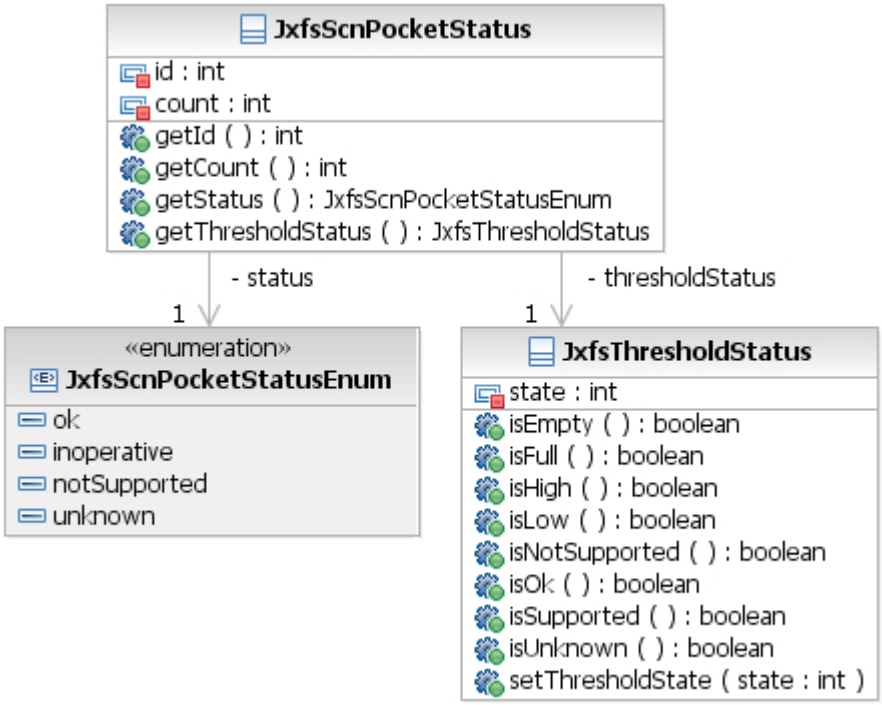
JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *mediaLastOperation* is less than 0.

6.1.19 JxfsScnPocketStatus

6.1.19.1 Usage

This class defines a pocket where media can be located after processing.

6.1.19.2 Class Hierarchy



6.1.19.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
id	*int*	R
count	*int*	R
status	*JxfsScnPocketStatusEnum*	R
thresholdStatus	*JxfsThresholdStatus*	R

Constructor	Parameter
JxfsScnPocketStatus	id
	count
	status
	thresholdStatus

Method	Return
get*Property*	*Property*

6.1.19.4 Properties

6.1.19.4.1 id

Type int
Remarks This property defines an unique numerical identifier for the pocket that will let the application distinguish from the rest of pockets.

6.1.19.4.2 count

Type int
Remarks Indicates the number of items in the pocket..

6.1.19.4.3 status

Type *JxfsScnPocketStatusEnum*
Remarks Indicates current status for this pocket.

6.1.19.4.4 thresholdStatus

Type JxfsThresholdStatus
Remarks Indicates current threshold status for this pocket.

6.1.19.5 Constructors

6.1.19.5.1 JxfsScnPocketStatus

Syntax ***public JxfsScnPocketStatus (int id, int count, JxfsScnPocketStatusEnum status, JxfsThresholdStatus thresholdStatus) throws JxfsException***
Exceptions Exceptions, which can be generated by this method.

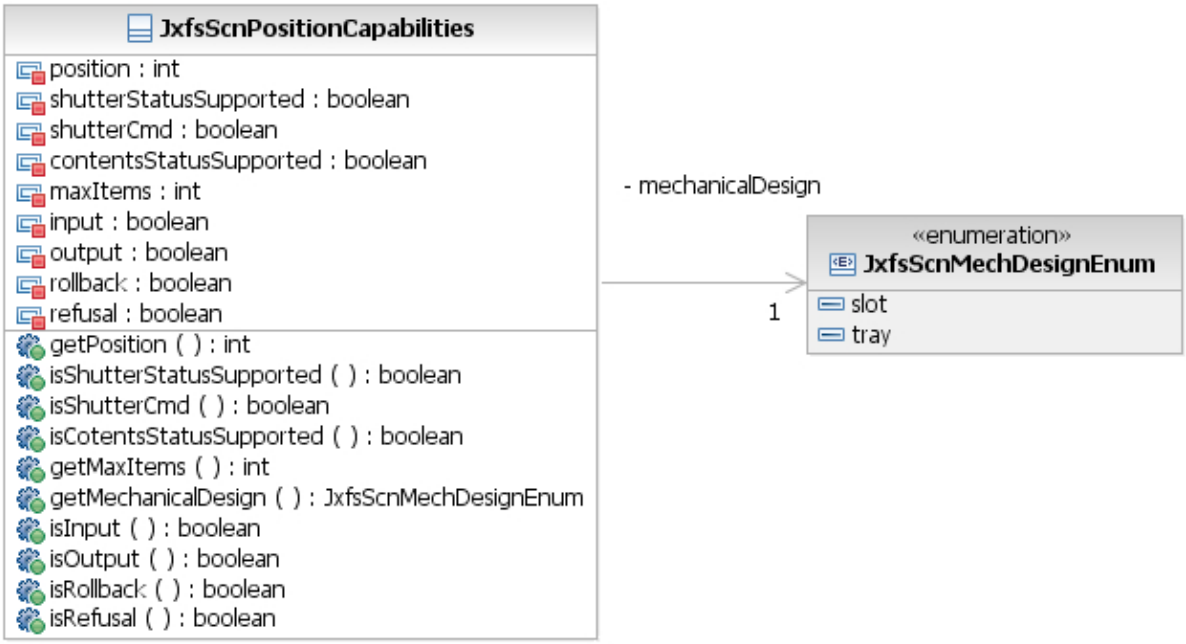
JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *status* is a null reference. • *thresholdStatus* is a null reference. • *count* less than zero (0)

6.1.20 JxfsScnPositionCapabilities

6.1.20.1 Usage

This class defines the positions features of the scanner device. The number of supported positions may change when the device is contacted.

6.1.20.2 Class Hierarchy



6.1.20.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
position	*int*	R
shutterStatusSupported	*boolean*	R
shutterCmd	*boolean*	R
contentsStatusSupported	*boolean*	R
maxItems	*int*	R
mechanicalDesign	*JxfsScnMechDesignEnum*	R
input	boolean	R
output	boolean	R
rollback	boolean	R
refusal	boolean	R

Constructor	Parameter
JxfsScnPositionCapabilities	position
	shutterStatusSupported
	shutterCmd
	contentsStatusSupported
	maxItems
	mechanicalDesign
	input
	output
	rollback
	refusal

Method	Return
get*Property*	*Property*
is*Property*	*boolean*

6.1.20.4 Properties

6.1.20.4.1 position

Type int

Remarks Identifies the position whose capabilities are provided. For more information about position definition refer to the Position Codes chapter.

Each element in the JxfsPositionCapabilities array reported by *positionsCapabilities* property should contain a unique value for this property representing a single position code list.

6.1.20.4.2 shutterStatusSupported

Type boolean

Remarks Specifies whether shutter status is supported for this position. When this property is *false* the corresponding *isNotSupported* query will return *true*.

6.1.20.4.3 shutterCmd

Type boolean

Remarks Defines if the shutter has to be explicitly controlled by the application. When *true*, the application is responsible for opening and closing the shutter using shutterMove.

If this property is *true* for an output position, then the *autoPresent* capability must be *notSupported*, as it would not be possible for the calling application to determine when it should open the dispense shutter, due to the possibility for a dispense to be delayed.

Even if *shutterCmd* is true a device service may close the shutter automatically. In this case a further close command of the application will return with JXFS_RC_SUCCESSFUL.

6.1.20.4.4 contentsStatusSupported

Type boolean

Remarks Specifies whether there is a sensor to detect if the position is empty. When this property is *false*, the corresponding *isNotSupported* query will return *true*.

6.1.20.4.5 maxItems

Type int

Remarks Maximum number of items which this position can hold. This is not a guaranteed value. It's an estimation of the number of items that can be held under normal conditions.

6.1.20.4.6 mechanicalDesign

Type *JxfsScnMechDesignEnum*

Remarks Specifies the mechanical design of this position.
For more details on the different position designs see chapter *Position Mechanical Design Notes* in *CWA Part 5: Cash Dispenser, Recycler and ATM Device Class Interface*.

6.1.20.4.7 input

Type boolean

Remarks Specifies whether this position can be used as source for: *scan*(), *process*() and *processBundle*() commands.

6.1.20.4.8 output

Type boolean

Remarks Specifies whether this position can be used as target for output items.

6.1.20.4.9 rollback

Type	boolean
Remarks	Specifies whether this position can be used as target for rollback() command.

6.1.20.4.10 refusal

Type	boolean
Remarks	Specifies whether refused items can be moved to this position during: *scan*(), *process*() and *processBundle*() commands.

6.1.20.5 Constructors

6.1.20.5.1 JxfsScnPositionCapabilities

Syntax	***public JxfsScnPositionCapabilites(int position, boolean shutterStatusSupported, boolean shutterCmd, boolean contentsStatusSupported, int maxItems, JxfsScnMechDesignEnum mechanicalDesign, boolean input, boolean output, boolean rollback, boolean refusal) throws JxfsException***
Exceptions	Exceptions, which can be generated by this method. JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies: • *position* is not one of the allowed position definition codes. For more information about position definition refer to the Position Codes chapter. • *maxItems* if equal or less than zero (0). • *mechanicalDesign* is a null reference.

6.1.21 JxfsScnPositionStatus

6.1.21.1 Usage

This class provides the status of the positions as well as the status of the modules related to it.

6.1.21.2 Class Hierarchy

6.1.21.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
position	*int*	R
contentsStatus	*JxfsScnContentsStatusEnum*	R
shutterStatus	*JxfsScnShutterStatus*	R
processingProblems	*JxfsScnPositionProcessingProblemsEnum*	R

Constructor	Parameter
JxfsScnPositionStatus	position
	contentsStatus
	shutterStatus
	processingProblems

Method	Return
get*Property*	*Property*

6.1.21.4 Properties

6.1.21.4.1 position

Type *int*
Remarks Indicates the position.

6.1.21.4.2 contentsStatus

Type *JxfsScnContentsStatusEnum*
Remarks Indicates the current state of the position.

6.1.21.4.3 shutterStatus

Type *JxfsScnShutterStatus*
Remarks Indicates the state of the position shutter.

6.1.21.4.4 processingProblems

Type *JxfsScnPositionProcessingProblemsEnum*
Remarks Indicates the status of the media in the position transport.

6.1.21.5 Constructors

6.1.21.5.1 JxfsScnPositionStatus

Syntax ***public JxfsScnPositionStatus(int position, JxfsScnContentsStatusEnum contentsStatus, JxfsScnShutterStatus shutterStatus, JxfsScnPositionProcessingProblemsEnum* processingProblems*) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

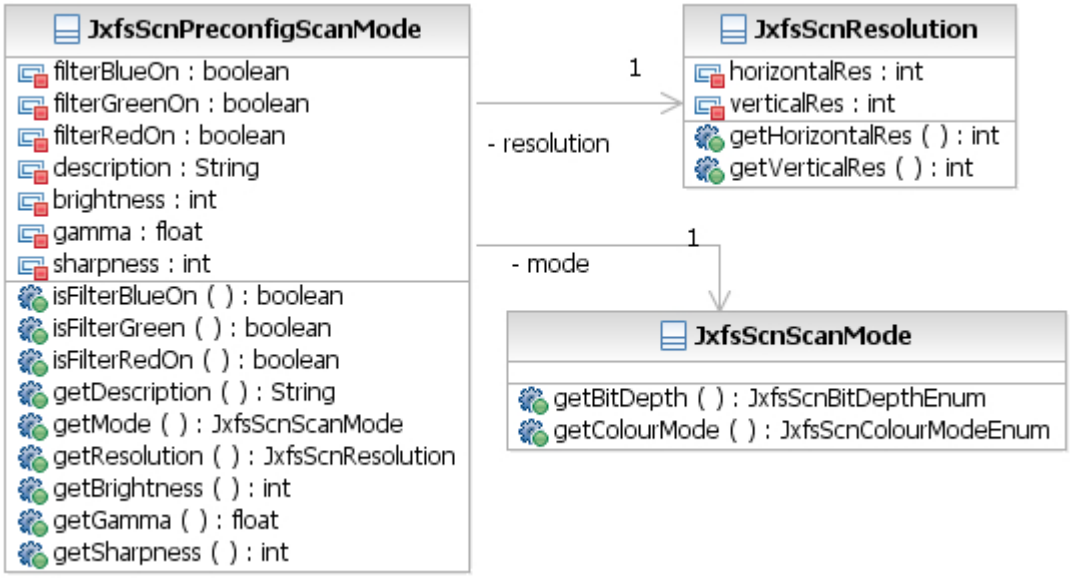
JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *position* is not one of the allowed position definition codes. For more information about position definition refer to the Position Codes chapter. • *contentsStatus* is a null reference. • *shutterStatus* is a null reference. • *processingProblems* is a null reference.

6.1.22 JxfsScnPreconfigScanMode

6.1.22.1 Usage

This class provides a predefined scan mode to the application in order to ease the configuration. It defines a complete set of settings to scan data.

6.1.22.2 Class Hierarchy



6.1.22.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
filterBlueOn	*boolean*	R
filterGreenOn	*boolean*	R
filterRedOn	*boolean*	R
description	*java.lang.String*	R
mode	*JxfsScnScanMode*	R
resolution	*JxfsScnResolution*	R
brightness	*int*	R
gamma	*float*	R
sharpness	*int*	R

Constructor	Parameter
JxfsScnPreconfigScanMode	filterBlueOn
	filterGreenOn
	filterRedOn
	description
	mode
	resolution
	brightness
	gamma
	sharpness

Method	Return
get*Property*	*Property*
is*Property*	*boolean*

6.1.22.4 Properties

6.1.22.4.1 filterBlueOn

Type boolean
Remarks Indicates if hardware must apply a blue light filter when scanning images

6.1.22.4.2 filterGreenOn

Type boolean
Remarks Indicates if hardware must apply a green light filter when scanning images.

6.1.22.4.3 filterRedOn

Type boolean
Remarks Indicates if hardware must apply a red light filter when scanning images.

6.1.22.4.4 description

Type java.lang.String
Remarks Contains the description for this whole set of settings.

6.1.22.4.5 mode

Type *JxfsScnScanMode*
Remarks Contains the scan mode used by this preconfigured set.

6.1.22.4.6 resolution

Type *JxfsScnResolution*
Remarks Contains the resolution used by this preconfigured set.

6.1.22.4.7 brightness

Type int
Remarks Indication of the brightness used by this preconfigured set as a percentage. It can be considered as the amount of light coming from the scanner.
If not supported, as indicated by the *IJxfsImageScanner.capabilities.brightnessControl* property, equals JXFS_C_SCN_NOT_SUPPORTED.

6.1.22.4.8 gamma

Type float
Remarks Indication of the gamma used by this preconfigured set.
The gamma correction is an adjustment to the light intensity of the scanner in order to match the output more closely to the original image. It is defined by the following power law expresion between the input and the output light:

$$V_{out} = V_{in}^{\gamma}$$

If not supported, as indicated by the *IJxfsImageScanner.capabilities.gammaControl* property, equals JXFS_C_SCN_NOT_SUPPORTED.

6.1.22.4.9 sharpness

Type int
Remarks Indication of the sharpness used by this preconfigured set as a percentage. This is a numerical measure that represents the image definition. If not supported, as indicated by the *IJxfsImageScanner.capabilities.sharpnessControl* property, equals JXFS_C_SCN_NOT_SUPPORTED.

6.1.22.5 Constructors

6.1.22.5.1 JxfsScnPreconfigScanMode

Syntax ***public JxfsScnPreconfigScanMode (boolean filterBlueOn, boolean filterGreenOn, boolean filterRedOn, java.lang.String description, JxfsScnScanMode mode, JxfsScnResolution resolution, int brightness, float gamma, int sharpness) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies:

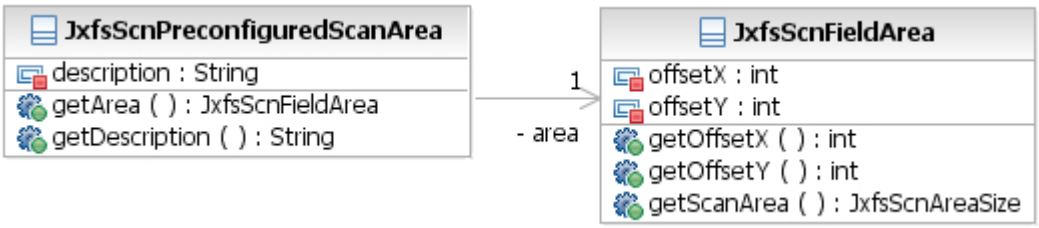
- *description* is a null reference.
- *mode* is a null reference.
- *resolution* is a null reference.

6.1.23 JxfsScnPreconfiguredScanArea

6.1.23.1 Usage

This class provides a predefined media size. It contains a description and information about the preconfigured area.

6.1.23.2 Class Hierarchy



6.1.23.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
area	*JxfsScnFieldArea*	R
description	*java.lang.String*	R

Constructor	Parameter
JxfsScnPreconfiguredScanArea	area
	description

Method	Return
get*Property*	*Property*

6.1.23.4 Properties

6.1.23.4.1 area

Type *JxfsScnFieldArea*
Remarks Contains information about the area that defines the media size.

6.1.23.4.2 description

Type java.lang.String
Remarks Contains the description for this preconfigured media size.

6.1.23.5 Constructors

6.1.23.5.1 JxfsScnPreconfiguredScanArea

Syntax ***public JxfsScnPreconfiguredScanArea(JxfsScnFieldArea, area, java.lang.String description) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies:

- area is a null reference.
- description is a null reference.

6.1.24 JxfsScnProcessData

6.1.24.1 Usage

This class provides properties to specify which type of process should be applied to the current media.

6.1.24.2 Class Hierarchy

6.1.24.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
encodeData	*java.lang.String*	R/W
encodeFont	*java.lang.String*	R/W
endorseFront	*boolean*	R/W
endorseRear	*boolean*	R/W
endorseDataFront	*java.util.List* of *JxfsScnEndorserData*	R/W
endorseDataRear	*java.util.List* of *JxfsScnEndorserData*	R/W
scanOrder	*JxfsScnScanOrderEnum*	R/W
stampFront	*boolean*	R/W
stampRear	*boolean*	R/W
stampFrontX	*int*	R/W
stampFrontY	*int*	R/W
stampRearX	*int*	R/W
stampRearY	*int*	R/W
pocket	*int*	R/W
position	*int*	R/W
itemId	*int*	R/W

Constructor	Parameter
JxfsScnProcessData	encodeData
	encodeFont
	endorseFront
	endorseRear
	endorseDataFront
	endorseDataRear
	scanOrder
	stampFront
	stampRear
	stampFrontX
	stampFrontY
	stampRearX
	stampRearY
	pocket
	position
	itemId

Method	Return
get*Property*	*Property*
is*Property*	*boolean*
set*Property*	*void*

6.1.24.4 Properties

6.1.24.4.1 encodeData

Type *java.lang.String*

Remarks Contains the data to be encoded. This property has meaning only if the *IJxfsScnCommonControl.capabilities.encoderCapabilities.encoderSupported is supported*. If *notSupported* then an empty String will be applied.

6.1.24.4.2 encodeFont

Type *java.lang.String*

Remarks Contains the font to be used when encoding. The font indicated must be one of *the IJxfsScnCommonControl.capabilities.encoderCapabilities.fonts* array.

6.1.24.4.3 endorseFront

Type *boolean*

Remarks Specifies whether the cheque must be endorsed at the front page or not. This property has meaning only if the *IJxfsScnCommonControl.capabilities.endorserCapabilities.endorserSupported* indicates supported front endorsing.

6.1.24.4.4 endorseRear

Type *boolean*

Remarks Specifies whether the cheque must be endorsed at the rear page or not. This property has meaning only if the *IJxfsScnCommonControl.capabilities.endorserCapabilities.endorserSupported* indicates supported rear endorsing.

6.1.24.4.5 endorseDataFront

Type *java.util.List* of *JxfsScnEndorserData* objects

Remarks Contains the data required for endorsement on the front side of media. The list may have as many *JxfsScnEndorserData* objects as indicated in the *IJxfsScnCommonControl.capabilities.endorserCapabilities.numberOfLines* property. If no data must be endorsed the list should be empty.
This property will only have meaning if *JxfsScnEndorserCapabilities.endorserSupported* indicates that front endorsing is supported.

6.1.24.4.6 endorseDataRear

Type *java.util.List* of *JxfsScnEndorserData* objects

Remarks Contains the data required for endorsement on the rear side of media. The list may have as many *JxfsScnEndorserData* objects as indicated in the *IJxfsScnCommonControl.capabilities.endorserCapabilities.numberOfLines* property. If no data must be endorsed the list should be empty.
This property will only have meaning if *JxfsScnEndorserCapabilities.endorserSupported* indicates that rear endorsing is supported.

6.1.24.4.7 scanOrder

Type *JxfsScnScanOrderEnum*

Remarks Specifies whether image data should be aquired in addition to processing the media, as well as the image must be acquired before, after or both the additional process. The *IJxfsScnCommonControl.capabilities.scanDuringProcessingSupported* capability controls when this property can be used.

6.1.24.4.8 stampFront

Type *boolean*

Remarks Specifies whether the media must be stamped at the front page or not. This property has meaning only if the *IJxfsScnCommonControl.capabilities.stampCapabilities.supported* indicates supported front stamping.

6.1.24.4.9 stampRear

Type *boolean*

Remarks Specifies whether the media must be stamped at the rear page or not. This property has meaning only if the *IJxfsScnCommonControl.capabilities.stampCapabilities.supported* indicates supported rear stamping.

6.1.24.4.10 stampFrontX

Type *int*

Remarks Specifies the horizontal position for stamping (if selectable) of the front side, from the left hand side of the media. The value is specified using *IJxfsScnCommonControl.capabilities.lengthUnit* and *IJxfsScnCommonControl.capabilities.unitBase* properties and is always positive.The property has a value from 0 to *JxfsScnCapabilities.chequeCapabilities.maxStampX*. If stamp position is not selectable this value should be JXFS_C_SCN_VALUE_NOT_INITIALIZED.

6.1.24.4.11stampFrontY

Type *int*

Remarks Specifies the vertical position for stamping (if selectable) of the front side, from the top of the media. The value is specified using *IJxfsScnCommonControl.capabilities.lengthUnit* and *IJxfsScnCommonControl.capabilities.unitBase* properties and is always positive. The property has a value from 0 to *JxfSCNCapabilities.chequeCapabilities.maxStampY.* If stamp position is not selectable this value should be JXFS_C_SCN_VALUE_NOT_INITIALIZED.

6.1.24.4.12stampRearX

Type *int*

Remarks Specifies the horizontal position for stamping (if selectable) of the rear side, from the top of the media. The value is specified using *IJxfsScnCommonControl.capabilities.lengthUnit* and *IJxfsScnCommonControl.capabilities.unitBase* properties and is always positive. The property has a value from 0 to *JxfsScnCapabilities.chequeCapabilities.maxStampY.* If stamp position is not selectable this value should be JXFS_C_SCN_VALUE_NOT_INITIALIZED.

6.1.24.4.13stampRearY

Type *int*

Remarks Specifies the vertical position for stamping (if selectable) of the rear side, from the top of the media. The value is specified using *IJxfsScnCommonControl.capabilities.lengthUnit* and *IJxfsScnCommonControl.capabilities.unitBase* properties and is always positive. The property has a value from 0 to *JxfsScnCapabilities.chequeCapabilities.maxStampY.* If stamp position is not selectable this value should be JXFS_C_SCN_VALUE_NOT_INITIALIZED.

6.1.24.4.14pocket

Type *int*

Remarks Specifies the destination pocket, from the available pockets indicated by the *IJxfsScnCommonControl.capabilities.pocketsId* property.

If *IJxfsScnCommonControl.capabilities.escrowSupported* is *notSupported*, after the processing, the media will be transported to the specified pocket. On the other hand, if *IJxfsScnCommonControl.capabilities.escrowSupported* is *supported* the media will be placed in the escrow (if pocket equals JXFS_C_SCN_ESCROW) or to the specified pocket (if pocket stands for a valid pocket id).

If the media can't be placed in the specified pocket, it will be returned to the reject position. If the reject position is not supported or inoperative it will be returned to the output position.

If pocket equals JXFS_C_SCN_VALUE_NOT_INITIALIZED use position property value.

6.1.24.4.15position

Type *int*

Remarks Identifies the destination position if pocket stands for JXFS_C_SCN_VALUE_NOT_INITIALIZED.

6.1.24.4.16itemId

Type *int*

Remarks Identifies the item to be processed with this JxfsScnProcessData object.

If the items are obtained from an input position this value should be JXFS_C_SCN_VALUE_NOT_INITIALIZED.

If the items are obtained from the escrow this value must be one of the ID sent along with the JXFS_I_SCN_DATA_AVAILABLE events of the command that stored the items in the escrow.

For a detailed description of the AutoFeed capability see chapter 4.3.

6.1.24.5 Constructors

6.1.24.5.1 JxfsScnProcessData

Syntax ***public JxfsScnProcessData (java.lang.String encodeData, java.lang.String encodeFont, boolean endorseFront, boolean endorseRear, java.util.List endorseDataFront, java.util.List endorseDataRear, JxfsScnScanOrderEnum scanOrder,boolean stampFront, boolean stampRear,int stampFrontX,int stampFrontY, int stampRearX,int stampRearY, int pocket, int position, int itemId) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

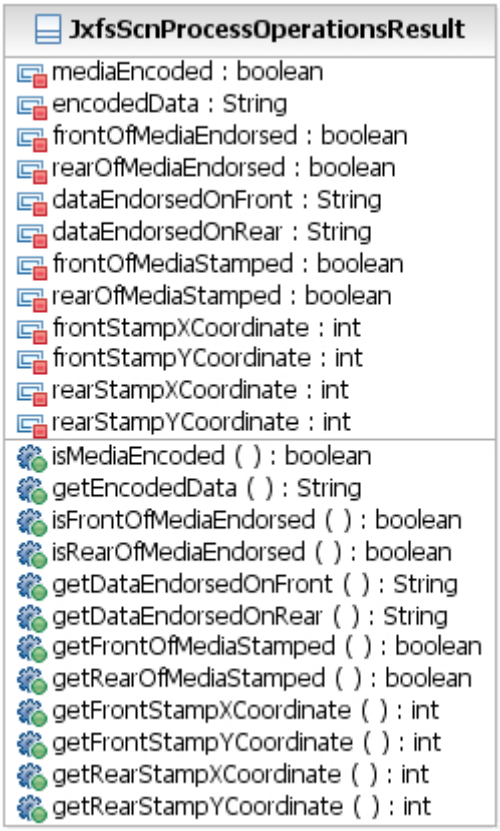
JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *encodeData* is a null reference. • *encodeFont* is a null reference. • *endorseDataFront* is a null reference or list objects with wrong types. • *endorseDataRear* is a null reference or list objects with wrong types. • *scanOrder* is a null reference. • *stamps* positions with negative values different from JXFS_C_SCN_VALUE_NOT_INITIALIZED • *pocket* is a negative and different from JXFS_C_SCN_VALUE_NOT_INITIALIZED and JXFS_C_SCN_ESCROW.

6.1.25 JxfsScnProcessOperationsResult

6.1.25.1 Usage

This class provides the results of the encoding, endorsing and stamping operations.

6.1.25.2 Class Hierarchy



6.1.25.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
mediaEncoded	*boolean*	R
encodedData	*java.lang.String*	R
frontOfMediaEndorsed	*boolean*	R
rearOfMediaEndorsed	*boolean*	R
dataEndorsedOnFront	*java.lang.String*	R
dataEndorsedOnRear	*java.lang.String*	R
frontOfMediaStamped	*boolean*	R
rearOfMediaStamped	*boolean*	R
frontStampXCoordinate	*int*	R
frontStampYCoordinate	*int*	R
rearStampXCoordinate	*int*	R
rearStampYCoordinate	*int*	R

Constructor	Parameter
JxfsScnProcessOperationsResult	mediaEncoded
	encodedData
	frontOfMediaEndorsed
	rearOfMediaEndorsed
	dataEndorsedOnFront
	dataEndorsedOnRear
	frontOfMediaStamped
	rearOfMediaStamped
	frontStampXCoordinate
	frontStampYCoordinate
	rearStampXCoordinate
	rearStampYCoordinate

Method	Return
get*Property*	*Property*
is*Property*	*boolean*

6.1.25.4 Properties

6.1.25.4.1 mediaEncoded

Type boolean
Remarks Specifies if media was encoded.

6.1.25.4.2 encodedData

Type java.lang.String
Remarks Specifies the data of the encoding.

6.1.25.4.3 frontOfMediaEndorsed

Type boolean
Remarks Specifies if the front side of the media was endorsed.

6.1.25.4.4 rearOfMediaEndorsed

Type boolean
Remarks Specifies if the rear side of the media was endorsed.

6.1.25.4.5 dataEndorsedOnFront

Type java.lang.String
Remarks Specifies the data endorsed on the front side of the media.

6.1.25.4.6 dataEndorsedOnRear

Type java.lang.String
Remarks Specifies the data endorsed on the rear side of the media.

6.1.25.4.7 frontOfMediaStamped

Type boolean
Remarks Specifies if the front side of the media was stamped.

6.1.25.4.8 rearOfMediaStamped

Type boolean
Remarks Specifies if the rear side of the media was stamped.

6.1.25.4.9 frontStampXCoordinate

Type int
Remarks Specifies the X coordinate of the front stamping.

6.1.25.4.10 frontStampYCoordinate

Type int
Remarks Specifies the Y coordinate of the front stamping.

6.1.25.4.11 rearStampXCoordinate

Type int
Remarks Specifies the X coordinate of the rear stamping.

6.1.25.4.12 rearStampYCoordinate

Type int
Remarks Specifies the Y coordinate of the rear stamping.

6.1.25.5 Constructors

6.1.25.5.1 JxfsScnProcessOperationsResults

Syntax ***public JxfsScnProcessOperationsResult(boolean mediaEncoded, java.lang.String encodedData, boolean frontOfMediaEndorsed, boolean rearOfMediaEndorsed, java.lang.String dataEndorsedOnFront, java.lang.String dataEndorsedOnRear, boolean frontOfMediaStamped, boolean rearOfMediaStamped, int frontStampXCoordinate, int frontStampYCoordinate, int rearStampXCoordinate, int rearStampYCoordinate) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID Generated if one of the following cases applies:

- *encodedData* is a null reference.
- *dataEndorsedOnFront* is a null reference.
- *dataEndorsedOnRear* is a null reference.
- *frontStampXCoordinate* is a negative value.
- *frontStampYCoordinate* is a negative value.
- *rearStampXCoordinate* is a negative value.
- *rearStampYCoordinate* is a negative value.

6.1.26 JxfsScnProgress

6.1.26.1 Usage

This class is returned by the *JXFS_I_SCN_SCAN_PROGRESS* Intermediate Event to inform the application about the progress of the execution of an acquiring process.

6.1.26.2 Class Hierarchy

6.1.26.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
percentageCompleted	*int*	R

Constructor	Parameter
JxfsScnProgress	percentageCompleted

Method	Return
getProperty	*Property*

6.1.26.4 Properties

6.1.26.4.1 percentageCompleted

Type int

Remarks Indicates the percentage completed in the acquiring process initiated by *scan(), process() or processBundle()* methods associated. Its value will range from 0 to 100 per each media. This information will be useful when the scan process is slow enough.

6.1.26.5 Constructors

6.1.26.5.1 JxfsScnProgress

Syntax ***public JxfsScnProgress(int percentageCompleted) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID Generated if one of the following cases applies:

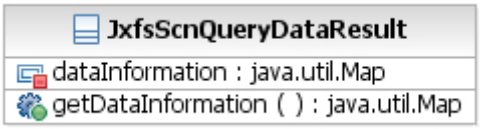
- *percentageCompleted* is less than 0.
- *percentageCompleted* is greater than 100.

6.1.27 JxfsScnQueryDataResult

6.1.27.1 Usage

This class holds the result of the *queryData* executions.

6.1.27.2 Class Hierarchy



6.1.27.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
dataInformation	*java.util.Map*	R

Constructor	Parameter
JxfsScnQueryDataResult	dataInformation

Method	Return
getProperty	*Property*

6.1.27.4 Properties

6.1.27.4.1 dataInformation

Type java.util.Map

Remarks Associative map composed by all the data identification numbers (represented *by java.lang.Integer* objects) as keys and data information (represented as *JxfsType* compatible object, refer to *The Acquiring Process* and *AutoFeed Capability* sections for more information on the type of object returned) as values.

6.1.27.5 Constructors

6.1.27.5.1 JxfsScnQueryDataResult

Syntax ***public JxfsScnQueryDataResult(java.util.Map dataInformation) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies:

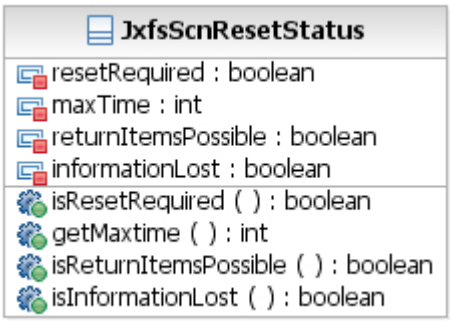
- *dataInformation* is a null reference.
- wrong data types in map

6.1.28 JxfsScnResetStatus

6.1.28.1 Usage

This class provides the information of the consequences of calling the *reset* method.

6.1.28.2 Class Hierrarcy



6.1.28.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
resetRequired	*boolean*	R
maxTime	*int*	R
returnItemsPossible	*boolean*	R
informationLost	*boolean*	R

Constructor	Parameter
JxfsScnResetStatus	resetRequired
	maxTime
	returnItemsPossible
	informationLost

Method	Return
isProperty	*boolean*
getProperty	*Property*

6.1.28.4 Properties

6.1.28.4.1 resetRequired

Type ***boolean***

Remarks If *true*, the hardware requires a reset command which will attempt to return it to a known operational state.

Normally, errors are resolved internally by the device service. There are, however, some scenarios in which this automatic recovery may not be performed:

- When automatic recovery will cause an observable impact on the customer. In this case, this method allows the application to decide the best time to perform the recovery.
- When automatic recovery will cause some valuable information to be lost (e.g. information required to deal with a customer dispute).
- When an unrecoverable error has occurred. In this case, the device has to be informed when the error is manually corrected, in order to allow it to perform any device specific activities required to return it to an operational state.

This property is set to *true* if and only if such exceptional events occur.

If a J/XFS call sends an operation complete event with result = JXFS_S_SCN_RESET_REQUIRED, the JxfsScnResetStatus.resetRequired property will always be *true*.

This property could be *true* without a previous operation complete event with result = JXFS_S_SCN_RESET_REQUIRED.

If this property is *true* and the device service is not closed or restarted, it will be *true* until a reset command is sent.

After calling reset, this property becomes *false* if the reset performed successfully and the device is operative again or *false* if the device requires manual intervention to be recovered.

6.1.28.4.2 maxTime

Type ***int***
Remarks Maximum estimated time to perform the reset, expressed in milliseconds.
A value of JXFS_C_SCN_UNKNOWN means unknown.

6.1.28.4.3 returnItemsPossible

Type ***boolean***
Remarks If *true*, the reset command may move items to a position accesible by the customer.

6.1.28.4.4 informationLost

Type ***boolean***
Remarks If *true*, the reset command may lose information during the execution and the counters or status could be inaccurate.

6.1.28.5 Constructors

6.1.28.5.1 JxfsScnResetStatus

Syntax ***public JxfsScnResetStatus(boolean resetRequired, int maxTime, boolean returnItemsPossible, boolean informationLost) throws JxfsException***
Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • maxTime is negative.

6.1.28.5.2 JxfsScnResetStatus

Syntax ***public JxfsScnResetStatus(boolean resetRequired, boolean returnItemsPossible, boolean informationLost)***
Exceptions No exception thrown.
Remarks Creates a JxfsScnResetStatus with unknown maxTime.

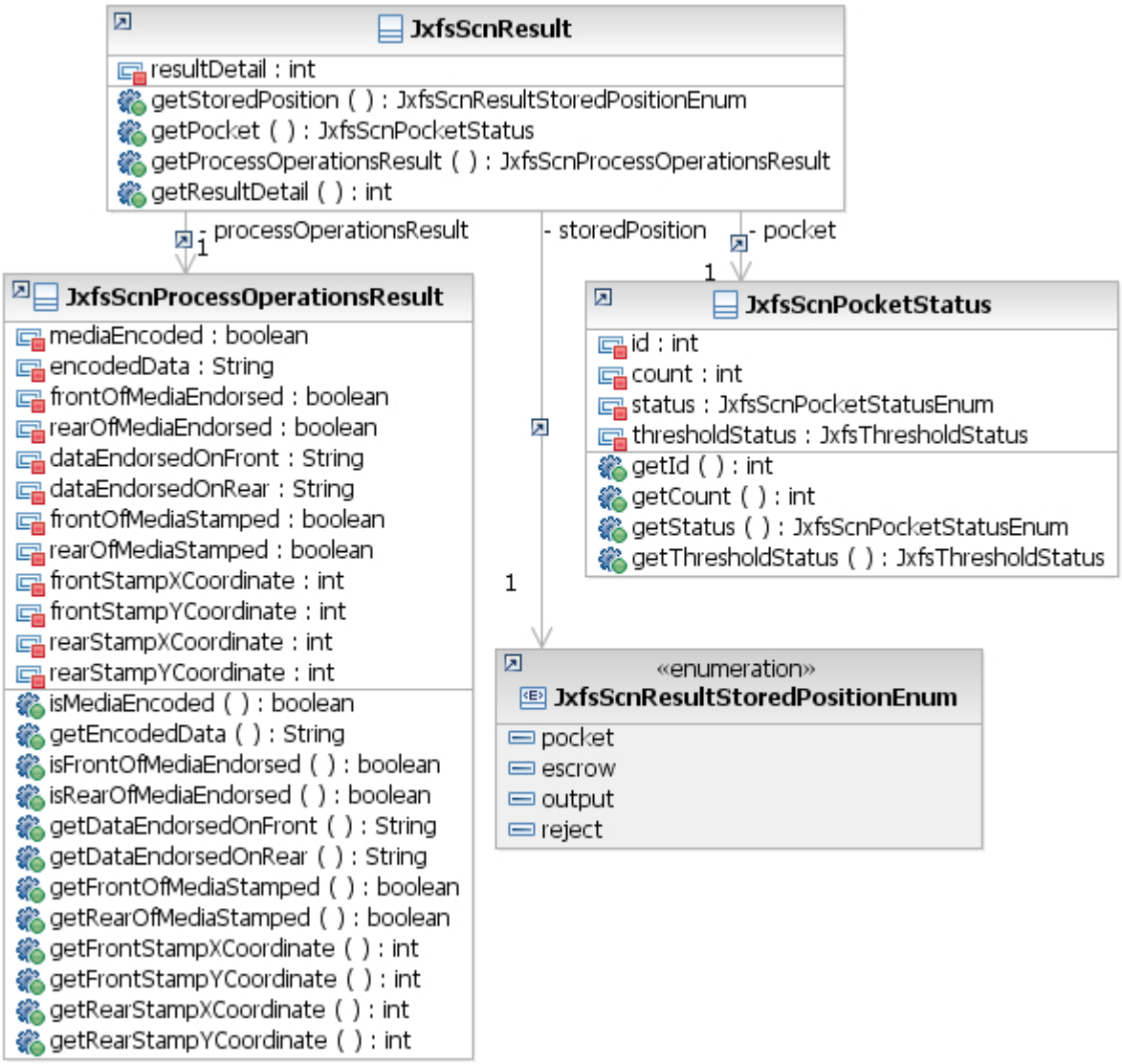
6.1.29 JxfsScnResult

6.1.29.1 Usage

Abstract basic class of the data returned by:

- *JxfsOperationCompleteEvent* event for *scan()*, *process()* and *processBundle()* operations.
- The values of the *JxfsScnQueryDataResult.dataInformation* map returned in the *JxfsOperationCompleteEvent* event for *queryData()*

6.1.29.2 Class Hierarchy



6.1.29.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
storedPosition	*JxfsScnResultStoredPositionEnum*	R
pocket	*JxfsScnPocketStatus*	R
processOperationsResult	*JxfsScnProcessOperationsResult*	R
resultDetail	*int*	R

Constructor	Parameter
JxfsScnResult	storedPosition
	pocket
	processOperationsResults
	resultDetail

Method	Return
get*Property*	*Property*

6.1.29.4 Properties

6.1.29.4.1 storedPosition

Type *JxfsScnResultStoredPositionEnum*
Remarks Specifies the position where the media is stored after scanning or processing.

6.1.29.4.2 pocket

Type *JxfsScnPocketStatus*
Remarks Specifies the pocket where the media is stored after scanning or processing.
This property will be null if storedPosition property is different from 'pocket'.

6.1.29.4.3 processOperationsResult

Type *JxfsScnProcessOperationsResult*
Remarks Specifies the results of the *process* operations (encoding, endorsing, stamping). It will be null if *JxfsScnCapabilities.additionalProcessingSupported* is *notSupported.*

6.1.29.4.4 resultDetail

Type int
Remarks This property will stand for JXFS_RC_SUCESSFUL if there's no additional information to be returned (i.e. item was properly scanned/processed).
In any other case gives further information about the result of the operation. It can take one of the values defined at the *error codes section*.

6.1.29.5 Constructors

6.1.29.5.1 JxfsScnResult

Syntax ***public JxfsScnResult(JxfsScnResultStoredPositionEnum storedPosition, JxfsScnPocketStatus pocket, JxfsScnProcessOperationsResult processOperationsResult, int resultDetail) throws JxfsException***
Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies:

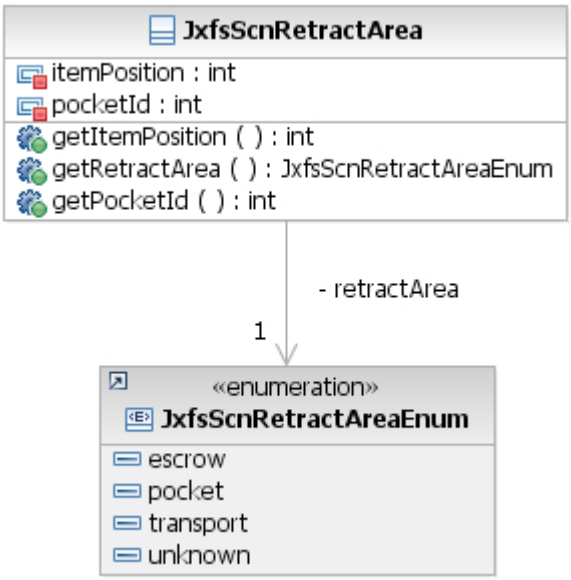
- *storedPosition* is null
- *pocket* is a null reference or less than zero (0) and *storedPosition* is not *pocket*.
- invalid code for *resultDetail*

6.1.30 JxfsScnRetractArea

6.1.30.1 Usage

Provides source and destination details for all items retracted by the device.

6.1.30.2 Class Hierarchy



6.1.30.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
itemPosition	*int*	R
retractArea	*JxfsScnRetractAreaEnum*	R
pocketId	*int*	R

Constructor	Parameter
JxfsScnRetractArea	itemPosition
	retractArea
	pocketId

Method	Return
getProperty	*Property*

6.1.30.3.1 itemPosition

Type *int*

Remarks Specifies the external positions from which to retract media Supported positions are determined from the *JxfsScnCapabilities.positionsCapabilities[].position* properties.

6.1.30.3.2 retractArea

Type *JxfsScnRetractAreaEnum*

Remarks Identifies the area to which items should or have been retracted. Supported areas are determined from the *JxfsScnCapabilities.supporterRetractAreas* property.

6.1.30.3.3 pocketId

Type *int*

Remarks If retractArea equals *JxfsScnRetractAreaEnum*.pocket this is the *JxfsScnPocketStatus.id* of the pocket into which items should or have been retracted. For a not valid pocketId, the device will determine which pocket to use.

6.1.30.4 Constructors

6.1.30.4.1 JxfsScnRetractArea

Syntax ***public JxfsScnRetractArea(int itemPosition, JxfsScnRetractAreaEnum retractArea, int pocketId) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID Generated if one of the following cases applies:

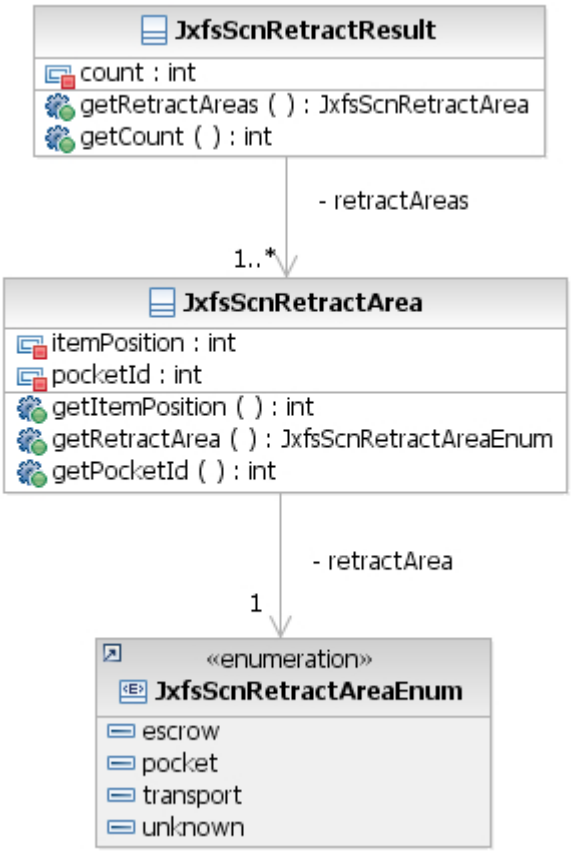
- *itemPosition* is not one of the allowed position definition codes. For more information about position definition refer to the Position Codes chapter.
- *retractArea* is a null reference.

6.1.31 JxfsScnRetractResult

6.1.31.1 Usage

Provides details of the *retract()* operation result.

6.1.31.2 Class Hierarchy



6.1.31.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
retractAreas	*java.util.List* of *JxfsScnRetractArea*	R
count	*int*	R

Constructor	Parameter
JxfsRetractResult	retractAreas
	count

Method	Return
getProperty	*Property*

6.1.31.3.1 retractAreas

Type *java.util.List* of *JxfsScnRetractArea*
Remarks Provides destination details for all the items retracted.

6.1.31.3.2 count

Type *int*
Remarks The amount of items that were retracted. It will be JXFS_C_SCN_NOT_SUPPORTED for those devices that are not able to count the retracted items.

6.1.31.4 Constructors

6.1.31.4.1 JxfsScnRetractResult

Syntax ***public JxfsScnRetractResult(java.util.List of JxfsScnRetractArea retractAreas, int count) throws JxfsException***
Exceptions Exceptions, which can be generated by this method.

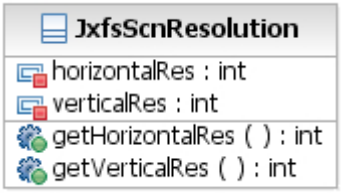
JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *retractAreas* is: null reference, empty or list objects with wrong types.

6.1.32 JxfsScnResolution

6.1.32.1 Usage

This class contains a valid resolution to acquire an image. A list of valid resolutions can be obtained through the *IJxfsImageScanner.imageCapabilities.resolutions* property. One of these objects is used in the *IJxfsImageScanner.configureImageScan* method prior to the scanning of an image.

6.1.32.2 Class Hierarchy



6.1.32.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
horizontalRes	*int*	R
verticalRes	*int*	R

Constructor	Parameter
JxfsScnResolution	horizontalRes
	verticalRes

Method	Return
get*Property*	*Property*

6.1.32.4 Properties

6.1.32.4.1 horizontalRes

Type int

Remarks Returns a valid horizontal resolution to be used in an acquiring process. Value indicated in DPI.

6.1.32.4.2 verticalRes

Type int

Remarks Returns a valid vertical resolution to be used in an acquiring process. Value indicated in DPI.

6.1.32.5 Constructors

6.1.32.5.1 JxfsScnResolution

Syntax ***public JxfsScnResolution(int horizontalRes, int verticalRes) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID Generated if one of the following cases applies:

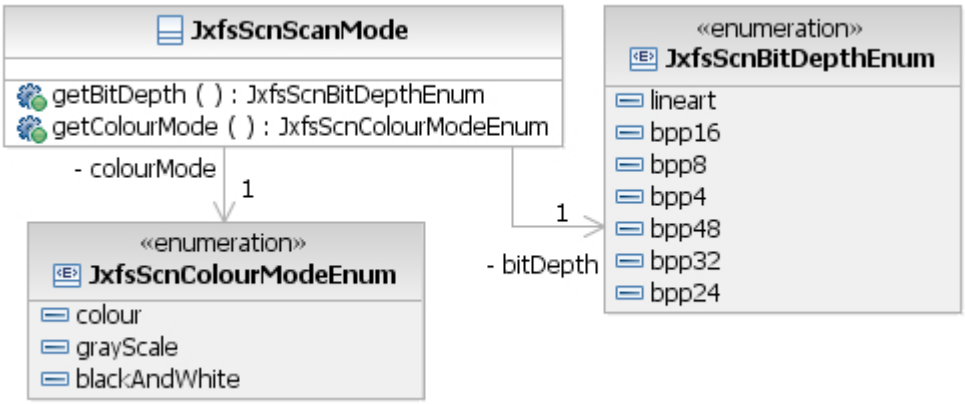
- Either *horizontalRes* or *verticalRes* are negative

6.1.33 JxfsScnScanMode

6.1.33.1 Usage

This class provides properties to specify the available scan colour modes and bit depths. An array containing a list of objects of this class can be retrieved from the *JxfsScnImageCapabilities* class.

6.1.33.2 Class Hierarchy



6.1.33.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
bitDepth	*JxfsScnBitDepthEnum*	R
colourMode	*JxfsScnColourModeEnum*	R

Constructor	Parameter
JxfsScnScanMode	bitDepth
	colourMode

Method	Return
get*Property*	*Property*

6.1.33.4 Properties

6.1.33.4.1 bitDepth

Type *JxfsScnBitDepthEnum*
Remarks Indicates an available bitDepth for image scanning.

6.1.33.4.2 colourMode

Type *JxfsScnColourModeEnum*
Remarks Indicates an available colour mode for image scanning.

6.1.33.5 Constructors

6.1.33.5.1 JxfsScnScanMode

Syntax ***public JxfsScnScanMode(JxfsScnBitDepthEnum bitDepth, JxfsScnColourModeEnum colourMode) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID — Generated if one of the following cases applies:

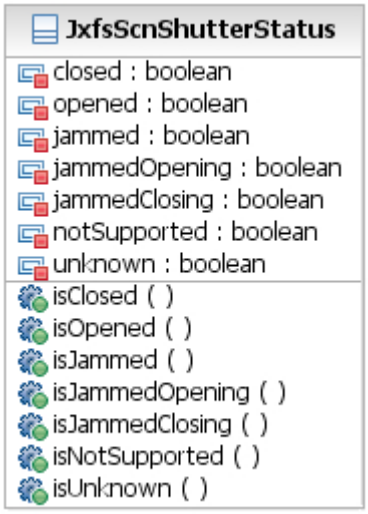
- *bitDepth* is a null reference.
- *colourMode* is a null reference.

6.1.34 JxfsScnShutterStatus

6.1.34.1 Usage

This support class defines the current status of a shutter module.

6.1.34.2 Class Hierarchy



6.1.34.3 Summary

Extends	Implements
JxfsType	

Query	Return
closed	boolean
opened	boolean
jammed	boolean
jammedOpening	boolean
jammedClosing	boolean
notSupported	boolean
unknown	boolean

Constructor	Parameter
JxfsScnShutterStatus	closed
	opened
	jammed
	jammedOpening
	jammedClosing
	notSupported
	unknown

Method	Return
is*Property*	*boolean*

6.1.34.4 Properties

6.1.34.4.1 closed

Type *boolean*
Remarks The shutter is closed.

6.1.34.4.2 opened

Type *boolean*
Default Value unknown
Remarks The shutter is opened.

6.1.34.4.3 jammed

Type *boolean*
Remarks One or more items are jammed in the shutter. This value will be *true* whenever the device detects a jam in the shutter. If device is able to report more precise information about this jam, *jammedOpening* or *jammedClosing* may be *true* as well.

6.1.34.4.4 jammedOpening

Type *boolean*
Remarks The shutter jammed while trying to open.If this value is *true*, then jammed property must be *true* and *jammedClosing* must be *false*.

6.1.34.4.5 jammedClosing

Type *boolean*
Remarks The shutter jammed while trying to close. If this value is *true*, then jammed property must be *true* and *jammedOpening* must be *false*.

6.1.34.4.6 notSupported

Type *boolean*
Remarks The position does not support shutter.

6.1.34.4.7 unkown

Type *boolean*
Remarks The state of the shutter is unknown.

6.1.34.5 Constructors

6.1.34.5.1 JxfsScnShutterStatus

Syntax ***public JxfsScnShutterStatus (boolean closed, boolean opened, boolean jammed, boolean jammedOpening, boolean jammedClosing, boolean notSupported, boolean unknown) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *opened* and *closed* are both *true*. • *jammed* is *false* and *jammedOpening* or *jammedClosing* are *true*. • *jammedOpening* and *jammedClosing* are both *true*. • *notSupported* is *true* and any of the rest are *true*. • *unknown* is *true* and any of the rest are *true*.

6.1.35 JxfsScnStampCapabilities

6.1.35.1 Usage

This support class defines the capabilities of a stamp module within the scanner device. The default object represents the object to be returned, if it is not (yet) known, what the device supports.

6.1.35.2 Class Hierarchy

6.1.35.3 Summary

Extends	Implements
JxfsType	

Property	Type	Default Value	Access
supported	*JxfsScnStampSupportEnum*	unknown	R
fixedHead	*JxfsScnFixedHeadEnum*	notSupported	R
maxStampX	*int*	JXFS_C_SCN_VALUE_NOT_INITIALIZED	R
maxStampY	*int*	JXFS_C_SCN_VALUE_NOT_INITIALIZED	R

Default Constructor	Return
JxfsScnStampCapabilities	Sets all properties to its default values

Constructor	Parameter
JxfsScnStampCapabilities	supported
	fixedHead
	maxStampX
	maxStampY

Method	Return
get*Property*	*Property*

6.1.35.4 Properties

6.1.35.4.1 supported

Type *JxfsScnStampSupportEnum*
Default Value unknown
Remarks Indicates supported stamping modes if any.

6.1.35.4.2 fixedHead

Type *JxfsScnFixedHeadEnum*
Default Value notSupported
Remarks Indicates if the stamp head can place stamps in user definable position or it is fixed in one specific place.

6.1.35.4.3 maxStampX

Type int
Default Value JXFS_C_SCN_VALUE_NOT_INITIALIZED
Remarks Indicates the maximum value for the X coordinate of the stamp head. The *JxfsScnProcessData.stampX* value can range from 0 to *maxStampX*. The value is specified using *IJxfsScnCommonControl.capabilities.lengthUnit* and *IJxfsScnCommonControl.capabilities.unitBase* properties and is always positive. If *fixedHead* property equals *completelyFixed* or *partiallyFixed* this property has no meaning and equals JXFS_C_SCN_VALUE_NOT_INITIALIZED.

6.1.35.4.4 maxStampY

Type int
Default Value JXFS_C_SCN_VALUE_NOT_INITIALIZED
Remarks Indicates the maximum value for the Y coordinate of the stamp head. The *JxfsScnProcessData.stampY* value can range from 0 to *maxStampY*. The value is specified using *IJxfsScnCommonControl.capabilities.lengthUnit* and *IJxfsScnCommonControl.capabilities.unitBase* properties and is always positive. If *fixedHead* property equals *completelyFixed* this property has no meaning and equals JXFS_C_SCN_VALUE_NOT_INITIALIZED.

6.1.35.5 Constructors

6.1.35.5.1 JxfsScnStampCapabilities

Syntax ***public JxfsScnStampCapabilities(JxfsScnStampSupportEnum supported, JxfsScnFixedHeadEnum fixedHead, int maxStampX, int maxStampY) throws JxfsException***
Exceptions Exceptions, which can be generated by this method.

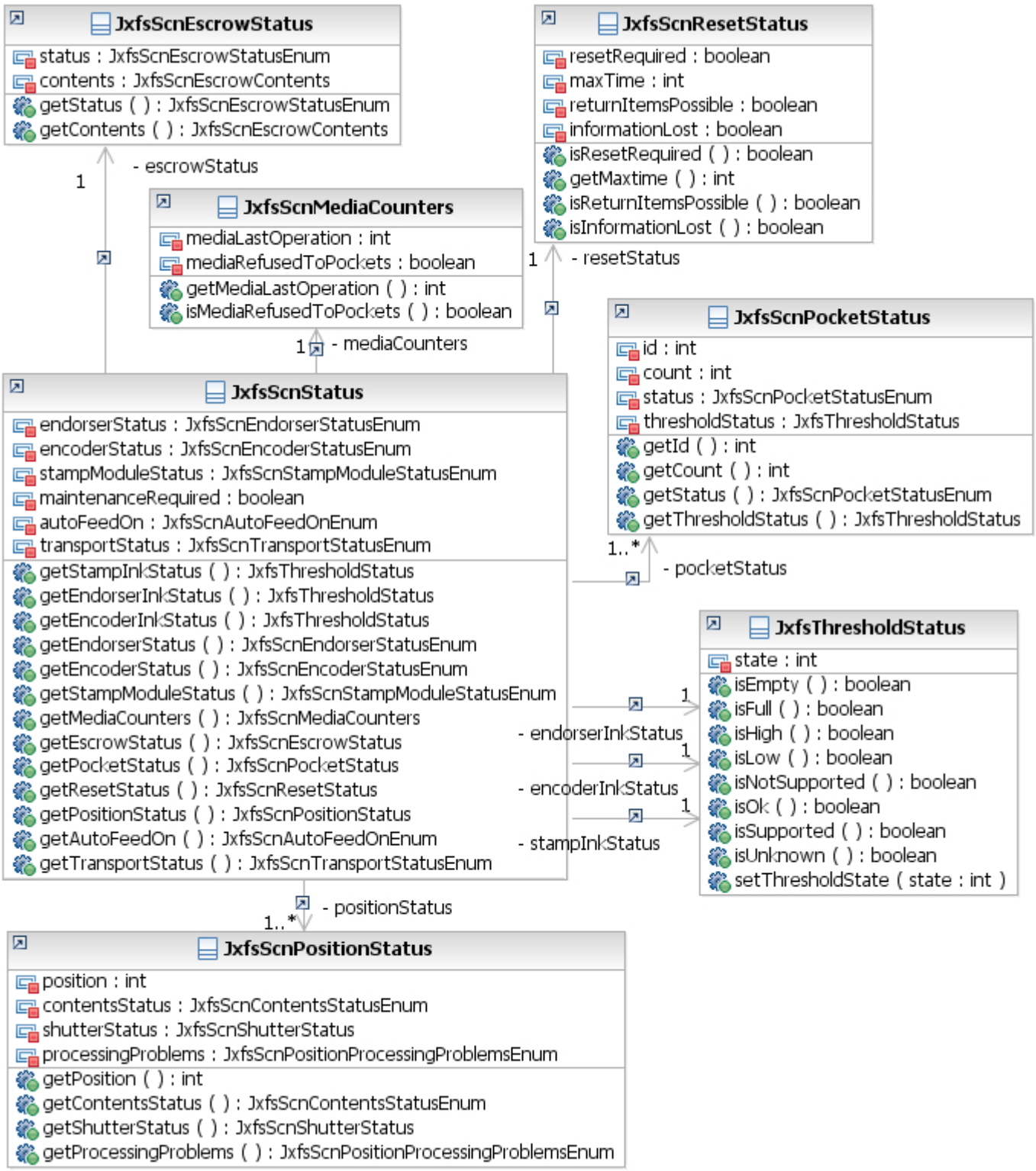
JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *supported* is a null reference. • *fixedHead* is a null reference. • negative values for *maxStamp* parameters

6.1.35.5.2 JxfsScnStampCapabilities

Syntax ***public JxfsScnStampCapabilities()***
Exceptions No additional exceptions thrown.

6.1.36 JxfsScnStatus

6.1.36.1 Class Hierarchy



6.1.36.2 Summary

Extends	Implements
JxfsType	

Property	Type	Access
stampInkStatus	*JxfsThresholdStatus*	R
endorserInkStatus	*JxfsThresholdStatus*	R
encoderInkStatus	*JxfsThresholdStatus*	R
endorserStatus	*JxfsScnEndorserStatusEnum*	R
encoderStatus	*JxfsScnEncoderStatusEnum*	R
stampModuleStatus	*JxfsScnStampModuleStatusEnum*	R
mediaCounters	*JxfsScnMediaCounters*	R
escrowStatus	*JxfsScnEscrowStatus*	R
pocketStatus	*java.util.List* of *JxfsScnPocketStatus*	R
resetStatus	*JxfsScnResetStatus*	R
positionStatus	*java.util.List* of *JxfsScnPositionStatus*	R
maintenanceRequired	*boolean*	*R*
autoFeedOn	*JxfsScnAutoFeedOnEnum*	*R*
transportStatus	*JxfsScnTransportStatusEnum*	*R*

Constructor	Parameter
JxfsScnStatus	stampInkStatus
	endorserInkStatus
	encoderInkStatus
	endorserStatus
	encoderStatus
	stampModuleStatus
	mediaCounters
	escrowStatus
	pocketStatus
	resetStatus
	positionStatus
	maintenanceRequired
	autoFeedOn
	transportStatus

Method	Return
get*Property*	*Property*

6.1.36.3 Properties

6.1.36.3.1 stampInkStatus

Type *JxfsThresholdStatus*

Remarks Specifies the status of the ink in the scanner stamp module. If no stamp is available, as indicated by the *IJxfsScnCommonControl.capabilities.stampCapabilities.supported* capability, the value of this status will be *unknown*.

Events If the value of this property changes, the Device Service will send all registered Status Listeners a Status Event with the following values:

Field	Value
Status	JXFS_S_SCN_STAMP_INK_STATUS *stampInkStatus* has changed.
Details	*JxfsThresholdStatus* object with the new stamp ink status.

6.1.36.3.2 endorserInkStatus

Type *JxfsThresholdStatus*

Remarks Specifies the status of the ink in the scanner endorser. If no endorser is available, as indicated by the *IJxfsScnCommonControl.capabilities.endorserCapabilities.endorserSupported* capability, the value of this status will be *unknown*.

Events If the value of this property changes, the Device Service will send all registered Status Listeners a Status Event with the following values:

Field	**Value**
Status	JXFS_S_SCN_ENDORSER_INK_STATUS *endorserInkStatus* has changed.
Details	*JxfsThresholdStatus* object with the new endorser ink status.

6.1.36.3.3 encoderInkStatus

Type *JxfsThresholdStatus*

Remarks Specifies the status of the ink in the scanner encoder. If no encoder is available, as indicated by the *IJxfsScnCommonControl.capabilities.encoderCapabilities.encoderSupported* capability, the value of this status will be *unknown*.

Events If the value of this property changes, the Device Service will send all registered Status Listeners a Status Event with the following values:

Field	**Value**
Status	JXFS_S_SCN_ENCODER_INK_STATUS *encoderInkStatus* has changed.
Details	*JxfsThresholdStatus* object with the new encoder ink status.

6.1.36.3.4 endorserStatus

Type *JxfsScnEndorserStatusEnum*

Remarks Specifies the general status of the scanner endorser module. If no endorser is available the value of this status will be *notSupported*.

Events If the value of this property changes, the Device Service will send all registered Status Listeners a Status Event with the following values:

Field	**Value**
Status	JXFS_S_SCN_ENDORSER_STATUS *endorserStatus* has changed.
Details	*JxfsScnEndorserStatusEnum* object with the new endorser status.

6.1.36.3.5 encoderStatus

Type *JxfsScnEncoderStatusEnum*

Remarks Specifies the general status of the scanner encoder module. If no encoder is available the value of this status will be *notSupported*.

Events If the value of this property changes, the Device Service will send all registered Status Listeners a Status Event with the following values:

Field	**Value**
Status	JXFS_S_SCN_ENCODER_STATUS *encoderStatus* has changed.
Details	*JxfsScnEncoderStatusEnum* object with the new encoder status

6.1.36.3.6 stampModuleStatus

Type *JxfsScnStampModuleStatusEnum*

Remarks Specifies the general status of the stamp module. If no stamp available the value of this status will be *notSupported*.

Events If the value of this property changes, the Device Service will send all registered Status Listeners a Status Event with the following values:

Field	**Value**
Status	JXFS_S_SCN_STAMP_STATUS *stampModuleStatus* has changed.
Details	*JxfsScnStampModuleStatusEnum* object with the new stamp module status

6.1.36.3.7 mediaCounters

Type *JxfsScnMediaCounters*

Remarks Specifies the state of the media.

Events If the value of this property changes, the Device Service will send all registered Status Listeners a Status Event with the following values:

Field	**Value**
Status	JXFS_S_SCN_MEDIA_COUNTERS mediaCounters has changed
Details	a *JxfsScnMediaCounters* object.

6.1.36.3.8 escrowStatus

Type *JxfsScnEscrowStatus*
Remarks Return the status information of the internal escrow module.
Events If the value of this property changes, the Device Service will send all registered Status Listeners a Status Event with the following values:

Field	Value
Status	JXFS_S_SCN_ESCROW_STATUS *escrowStatus* has changed.
Details	*JxfsScnEscrowStatus* with the new escrow status.

6.1.36.3.9 pocketStatus

Type *java.util.List* of *JxfsScnPocketStatus* objects
Remarks Return the status information of the supported pockets.
Events If the value of any of the *JxfsScnPocketStatus* changes, the Device Service will send all registered Status Listeners a Status Event with the following values:

Field	Value
Status	JXFS_S_SCN_POCKET_STATUS *pocketStatus* has changed.
Details	*JxfsScnPocketStatus* objects with the new pocket status.

6.1.36.3.10 resetStatus

Type *JxfsScnResetStatus*
Remarks Provides information on the consequences of calling the *reset* method.
Whenever the device needs to be *reset* for any reason a JXFS_S_SCN_RESET_REQUIRED Status Event is generated.

6.1.36.3.11 positionStatus

Type *java.util.List* of *JxfsScnPositionStatus*
Initial Value Empty List until a successful open has completed and the device is in working state.
Remarks Indicates the status of the positions as well as the status of the modules related with each position (transport, shutter,..)
Events If the value of this property changes, the Device Service will send all registered Status Listeners a Status Event with the following values:

Field	Value
Status	JXFS_S_SCN_POSITION_CHANGED *positionStatus* has changed.
Details	*JxfsScnPositionStatus* object with the new position status.

6.1.36.3.12 maintenanceRequired

Type *boolean*
Initial Value false
Remarks If the device encounters a problem that cannot be fixed by the device service and application alone, but needs the intervention of an operator (i.e. to remove some jammed items), this is indicated by this property. Unless the problem is fixed an application shall not call *reset* to prevent unnecessary device operation. After the problem has been solved by the operator, the application may issue a *reset* command to indicate to the device service that the original problem is solved and the device service may try to bring the device again in a working status.

true - the device service expects a *reset* as the indication that an operator intervention has been processed, the original problem solved and the device service may try again to set the device into an operable state. If this property is equal to *true*, the *resetRequired* property is always set to *false* to prevent endless resets by the application.

false - the device is in an operable state or it is expected that an operable state can be achieved by usage of the available API functionality.

Whenever the device needs an intervention by the operator a JXFS_S_SCN_MAINTENANCE_REQUIRED Status Event is generated.

6.1.36.3.13 autoFeedOn

Type *JxfsScnAutoFeedOnEnum*

Remarks Specifies the state of the automatic document feeder.

Events If the value of this property changes, the Device Service will send all registered Status Listeners a Status Event with the following values:

Field	Value
Status	JXFS_S_SCN_AUTOFEED_ON *autoFeedOn* has changed.
Details	*JxfsScnAutoFeedOnEnum* object with the new autoFeed value.

6.1.36.3.14 transportStatus

Type *JxfsScnTransportStatusEnum*

Remarks Indicates the status of the transport unit.

Events If the value of this property changes, the Device Service will send all registered Status Listeners a Status Event with the following values:

Field	Value
Status	JXFS_S_SCN_TRANSPORT_CHANGED *transportStatus* has changed.
Details	*JxfsScnTransportStatusEnum* object with the new transportStatus value.

6.1.36.4 Constructors

6.1.36.4.1 JxfsScnStatus

Syntax ***public JxfsScnStatus (JxfsThresholdStatus stampInkStatus, JxfsThresholdStatus endorserInkStatus, JxfsThresholdStatus encoderInkStatus, JxfsScnEndorserStatusEnum endorserStatus, JxfsScnEncoderStatusEnum encoderStatus, JxfsScnStampModuleStatusEnum stampModuleStatus, JxfsScnMediaCounters mediaCounters, JxfsScnEscrowStatus escrowStatus, java.util.List pocketStatus, JxfsScnResetStatus resetStatus, java.util.List positionStatus,boolean maintenanceRequired, JxfsScnAutoFeedOnEnum autoFeedOn, JxfsScnTransportStatusEnum transportStatus) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

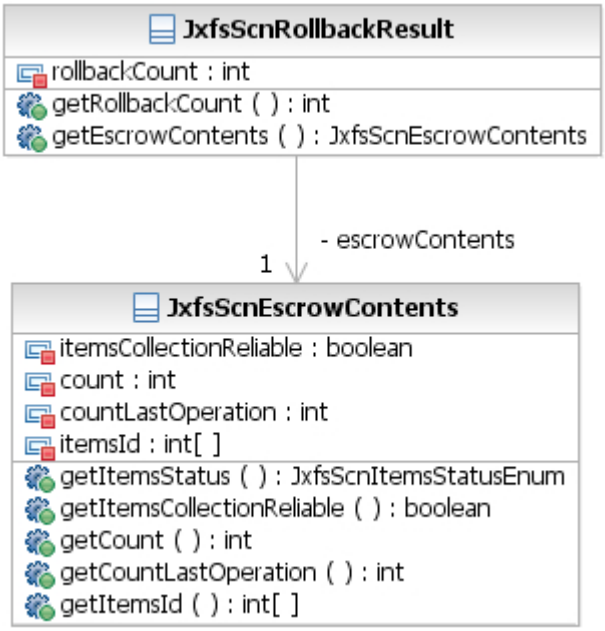
JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies:
	• null references for any of the following objects:
	  ○ *stampInkStatus*
	  ○ *endorserInkStatus*
	  ○ *encoderInkStatus*
	  ○ *endorserStatus*
	  ○ *encoderStatus*
	  ○ *stampModuleStatus*
	  ○ *mediaCounters*
	  ○ *escrowStatus*
	  ○ *pocketStatus*
	  ○ *resetStatus*
	  ○ *positionStatus*
	  ○ *autoFeedOn*
	  ○ *transportStatus*
	• List objects (*pocketStatus*, *positionstatus*) with wrong types.

6.1.37 JxfsScnRollbackResult

6.1.37.1 Usage

This contains the result for *rollback* operations.

6.1.37.2 Class Hierarchy



6.1.37.3 Summary

Extends	Implements
JxfsType	

Property	Type	Access
rollbackCount	*int*	R
escrowContents	*JxfsScnEscrowContents*	R

Constructor	Parameter
JxfsScnRollbackResult	rollBackCount
	escrowContents

Method	Return
get*Property*	*Property*

6.1.37.4 Properties

6.1.37.4.1 rollbackCount

Type int
Remarks The total number of medias moved from the internal escrow to the output position in the rollback operation.

6.1.37.4.2 escrowContents

Type *JxfsScnEscrowContents*
Remarks The contents of the escrow.

6.1.37.5 Constructors

6.1.37.5.1 JxfsScnRollbackResult

Syntax ***public JxfsScnRollbackResult (int rollbackCount, JxfsScnEscrowContents escrowContents) throws JxfsException***

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: • *rollbackCount* is negative. • *escrowContents* is a null reference.

7 Events

7.1 Intermediate Events

7.1.1 Intermediate Event Code Summary and Description

Code	Value	Meaning
15300	JXFS_I_SCN_NO_MEDIA_PRESENT	The operation request cannot progress because there is no media inserted.
15301	JXFS_I_SCN_MEDIA_INSERTED	The operation request can continue because a media item has been inserted.
15302	JXFS_I_SCN_SCAN_PROGRESS	It can be sent optionally from time to time for application use indicating the progress of the scanning operation. Depending on the device type the acquiring process can take some time and this event can help to create a friendlier user interface where information about how the acquiring process is proceeding is displayed.
15303	JXFS_I_SCN_DATA_AVAILABLE	It is sent to inform that some media has been scanned. Contains an id to retrieve the data of the scanning using the *queryData* method.
15304	JXFS_I_SCN_INPUT_REFUSED	It is sent with the first item refused.

7.1.2 IJxfsScnCommonControl Intermediate Events

Methods	*processBundle*	*process*	*scan*	*shutterMove*	*rollback*	*retract*
Intermediate Events						
JXFS_I_SCN_NO_MEDIA_PRESENT	X	X	X			
JXFS_I_SCN_MEDIA_INSERTED	X	X	X			
JXFS_I_SCN_SCAN_PROGRESS	X	X	X			
JXFS_I_SCN_DATA_AVAILABLE	X	X	X			
JXFS_I_SCN_INPUT_REFUSED	X	X	X			

7.1.3 Intermediate Event Details

7.1.3.1 JXFS_I_SCN_NO_MEDIA_PRESENT

The operations request cannot progress because there is no media present.

Field	Value
operationID	*operationID* of the method initiating this event
identificationID	*identificationID* of the method initiating this event.
reason	JXFS_I_SCN_NO_MEDIA_PRESENT
data	*null*

7.1.3.2 JXFS_I_SCN_MEDIA_INSERTED

The scan or process operation request continues because a media has been inserted.

Field	Value
operationID	*operationID* of the method initiating this event
identificationID	*identificationID* of the method initiating this event.
reason	JXFS_I_SCN_MEDIA_INSERTED
data	*null*

7.1.3.3 JXFS_I_SCN_SCAN_PROGRESS

It can be sent optionally from time to time for application use indicating the progress in the operation. The broadcasting of this Intermediate Event is dependant on the value of the *IJxfsScnCommonControl.capabilities.scanProgressSupported* capability. The issue rate will be defined by the specific device. Depending on the device type the acquiring process can take some time and this event can help creating a friendlier user interface where information about how the acquiring process is progressing can be displayed.

Field	Value
operationID	*operationID* of the method initiating this event
identificationID	*identificationID* of the method initiating this event.
reason	JXFS_I_SCN_SCAN_PROGRESS
data	*JxfsScnProgress* object

7.1.3.4 JXFS_I_SCN_DATA_AVAILABLE

It is sent to notify that some media has been scanned.
The order these events are sent is the same order in which they were scanned.

Field	Value		
operationID	*operationID* of the method initiating this event		
identificationID	*identificationID* of the method initiating this event.		
reason	JXFS_I_SCN_DATA_AVAILABLE		
data	**Data**	**Type**	**Description**
	scanId	*JxfsScnDataAvailable*	Holds the identification of the available item.

7.1.3.5 JXFS_I_SCN_INPUT_REFUSED

Sent whenever items are refused

Field	Value		
operationID	*operationID* of the method initiating this event		
identificationID	*identificationID* of the method initiating this event.		
reason	JXFS_I_SCN_INPUT_REFUSED		
data	**Data**	**Type**	**Description**
	position Status	*JxfsScnPositionStatus*	Status of the position with the refused items.

7.2 Status Events

7.2.1 Status Event Code Summary and Description

Code	Value	Meaning
15200	JXFS_S_SCN_POSITION_CHANGED	The status for one of the supported positions has changed.
15201	JXFS_S_SCN_ENDORSER_STATUS	*endorserStatus* property has changed.
15202	JXFS_S_SCN_ENCODER_STATUS	*encoderStatus* property has changed.
15203	JXFS_S_SCN_POCKET_STATUS	a pocket *status* property has changed.
15204	JXFS_S_SCN_ESCROW_STATUS	escrowStatus property has changed
15205	JXFS_S_SCN_ESCROW_ITEMS_CHANGED	the content of the escrow has changed
15206	JXFS_S_SCN_RESET_REQUIRED	a *reset* command should be performed to return the device into operational state.
15207	JXFS_S_SCN_STAMP_STATUS	*stampModuleStatus* property has changed.
15208	JXFS_S_SCN_STAMP_INK_STATUS	the status of the ink for the stamping module has changed.
15209	JXFS_S_SCN_ENDORSER_INK_STATUS	the status of the ink for the endorsing module has changed.
15210	JXFS_S_SCN_ENCODER_INK_STATUS	the status of the ink for the encoding module has changed.
15211	JXFS_S_SCN_MEDIA_COUNTERS	mediaCounters property has changed
15212	JXFS_S_SCN_MAINTENANCE_REQUIRED	maintenanceRequired property has changed
15213	JXFS_S_SCN_AUTOFEED_ON	the autoFeedOn status has changed
15214	JXFS_S_SCN_RESET_STATUS_CHANGED	the reset status has changed.
15215	JXFS_S_SCN_TRANSPORT_CHANGED	the transport status has changed

7.2.2 Status Event Details

7.2.2.1 JXFS_S_SCN_POSITION_CHANGED

This event is sent whenever *JxfsScnStatus.positionStatus[x]*.changes its state.

Field	Value
Status	JXFS_S_SCN_POSITION_CHANGED
Details	*JxfsScnPositionStatus* object. Status of the position that changes the state.

7.2.2.2 JXFS_S_SCN_ENDORSER_STATUS

This event is sent whenever the general status of the endorser subdevice changes.

Field	Value
Status	JXFS_S_SCN_ENDORSER_STATUS
Details	*JxfsScnEndorserStatusEnum* object.

7.2.2.3 JXFS_S_SCN_ENCODER_STATUS

This event is sent whenever the general status of the encoder subdevice changes.

Field	Value
Status	JXFS_S_SCN_ENCODER_STATUS
Details	*JxfsScnEncoderStatusEnum* object.

7.2.2.4 JXFS_S_SCN_POCKET_STATUS

This event is sent if the *status* property of any *JxfsScnPocketStatus* object changes to indicate the application the new status of the pocket.

Field	Value
Status	JXFS_S_SCN_POCKET_STATUS
Details	*JxfsScnPocketStatus* object that has changed its status.

7.2.2.5 JXFS_S_SCN_ESCROW_STATUS

This event is sent if the status of the escrow module has changed.

Field	Value
Status	JXFS_S_SCN_ESCROW_STATUS
Details	*JxfsScnEscrowStatusEnum* object.

7.2.2.6 JXFS_S_SCN_ESCROW_ITEMS_CHANGED

This event is sent if content in the escrow has changed.

Field	Value
Status	JXFS_S_SCN_ESCROW_ITEMS_CHANGED
Details	*JxfsScnEscrowStatus* object.

7.2.2.7 JXFS_S_SCN_RESET_REQUIRED

This event is sent when the device needs a *reset* operation in order to get it back to operational state.

Field	Value
Status	JXFS_S_SCN_RESET_REQUIRED
Details	*JxfsScnResetStatus* object

7.2.2.8 JXFS_S_SCN_STAMP_STATUS

This event is sent if the status of the stamp module has changed.

Field	Value
Status	JXFS_S_SCN_STAMP_STATUS
Details	*JxfsScnStampModuleStatusEnum* object

7.2.2.9 JXFS_S_SCN_STAMP_INK_STATUS

The status of the ink for the stamping module has changed.

Field	Value
Status	JXFS_S_SCN_STAMP_INK_STATUS
Details	*JxfsThresholdStatus* inkStatus object

7.2.2.10 JXFS_S_SCN_ENDORSER_INK_STATUS

The status of the ink for the endorsing module has changed.

Field	Value
Status	JXFS_S_SCN_ENDORSER_INK_STATUS
Details	*JxfsThresholdStatus* inkStatus object

7.2.2.11 JXFS_S_SCN_ENCODER_INK_STATUS

The status of the ink for the encoding module has changed.

Field	Value
Status	JXFS_S_SCN_ENCODER_INK_STATUS
Details	*JxfsThresholdStatus* inkStatus object

7.2.2.12 JXFS_S_SCN_MEDIA_COUNTERS

The status of the media counters has changed.

Field	Value
Status	JXFS_S_SCN_MEDIA_COUNTERS
Details	*JxfsScnMediaCounters* object

7.2.2.13 JXFS_S_SCN_MAINTENANCE_REQUIRED

Generated when either the device needs an intervention by an operator to continue normal execution or when the device does not need an intervention any more.

Field	Value
Status	JXFS_SCN_MAINTENANCE_REQUIRED
Details	*JxfsScnStatus* object

7.2.2.14 JXFS_S_SCN_AUTOFEED_ON

The autoFeed value has changed.

Field	Value
Status	JXFS_S_SCN_AUTOFEED_ON
Details	*JxfsScnAutoFeedOnEnum* object

7.2.2.15 JXFS_S_SCN_RESET_STATUS_CHANGED

This Status Event is sent, if the reset status changes.

Field	Value
Status	JXFS_S_SCN_RESET_STATUS_CHANGED
Details	*JxfsScnResetStatus* object.

7.2.2.16 JXFS_S_SCN_TRANSPORT_CHANGED

The transport status has changed.

Field	Value
Status	JXFS_S_SCN_TRANSPORT_CHANGED
Details	*JxfsScnTransportStatusEnum* object.

8 Codes

8.1 Error Codes

Code	Value	Description
15000	JXFS_E_SCN_NOMEDIA	There is no media to work on.
15001	JXFS_E_SCN_MEDIA_INVALID	No appropriate media was found.
15002	JXFS_E_SCN_MEDIA_JAMMED	Media is jammed.
15003	JXFS_E_SCN_MEDIA_TYPE_UNSUPPORTED	The media that the device is trying to scan or process is not supported.
15004	JXFS_E_SCN_SCAN_FAILURE	No scan conditions were satisfied.
15005	JXFS_E_SCN_SHUTTER_FAILED	The shutter failed to open/close.
15006	JXFS_E_SCN_POSITION_ERROR	An error has occured at an input/output position. *IJxfsScnCommonControl.positionStatus*' s, for those positions in use, should be queried to determine whether a hardware error has occurred or whether the position has simply become full.
15007	JXFS_E_SCN_STORAGE_FULL	Execution cannot proceed because one or more of the required pocket or escrow to perform the operation is full. The current status of the pockets and escrow should be checked to determine the cause of the problem.
15008	JXFS_E_SCN_STAMP_NOT_SUPPORTED	A stamping operation on the front or rear of the media cannot be performed.
15009	JXFS_E_SCN_ENCODE_NOT_SUPPORTED	An encoding operation on the media cannot be performed.
15010	JXFS_E_SCN_ENDORSE_NOT_SUPPORTED	An endorsing operation on the front or rear of the media cannot be performed.
15011	JXFS_E_SCN_INVALID_ENCODE_DATA	Data cannot be encoded on media.
15012	JXFS_E_SCN_INVALID_ENDORSE_DATA	Data cannot be endorsed on media.

8.2 Operation Codes

The following codes identify the operation that generated an *JxfsOperationCompleteEvent* or *JxfsIntermediateEvent*:

Code	Value	Method
15100	JXFS_O_SCN_SCAN	*scan*
15101	JXFS_O_SCN_PROCESS	*process*
15102	JXFS_O_SCN_PROCESS_BUNDLE	*processBundle*
15103	JXFS_O_SCN_CONFIGURE_CHEQUE_SCAN	*configureChequeScan*
15104	JXFS_O_SCN_CONFIGURE_IMAGE_SCAN	*configureImageScan*
15105	JXFS_O_SCN_RESET	*reset*
15106	JXFS_O_SCN_ROLLBACK	*rollback*
15107	JXFS_O_SCN_SHUTTER_MOVE	*shutterMove*
15108	JXFS_O_SCN_RETRACT	*retract*

9 Constants

Code	Value	Description
-1	JXFS_C_SCN_VALUE_NOT_INITIALIZED	The value has not been initialized.
-2	JXFS_C_SCN_ESCROW	Value used to identify escrow destination in commands requiring pockets (*scan*, process and processBundle).
-3	JXFS_C_SCN_NOT_SUPPORTED	The property is not supported.
-4	JXFS_C_SCN_UNKNOWN	The content of the property is not yet known.

9.1 Position Codes

These following codes are used to identify input, output and reject positions.

Code		Description
4	JXFS_C_SCN_POS_LEFT	left side
8	JXFS_C_SCN_POS_CENTER	center side
16	JXFS_C_SCN_POS_RIGHT	right side
32	JXFS_C_SCN_POS_FRONT	front side
64	JXFS_C_SCN_POS_REAR	rear side
128	JXFS_C_SCN_POS_TOP	top side
256	JXFS_C_SCN_POS_BOTTOM	bottom side

9.2 Barcode formats

The following list provides contants for reserved barcode formats.

Name		Description
“upca_e”	JXFS_C_SCN_BARCODE_UPCA_E	UPC-A/E barcode type.
“ean8_13”	JXFS_C_SCN_BARCODE_EAN8_13	EAN-8/13 barcode type.
“jan8_13”	JXFS_C_SCN_BARCODE_JAN8_13	JAN 8/13 barcode type.
“code39”	JXFS_C_SCN_BARCODE_CODE39	CODE 39 barcode type.
“code128”	JXFS_C_SCN_BARCODE_CODE128	CODE 128 barcode type.
“nw7”	JXFS_C_SCN_BARCODE_NW7	NW-7 (CODABAR) barcode type.
“itf”	JXFS_C_SCN_BARCODE_ITF	Interleaved 2 of 5 (ITF) barcode type.
“upca_e2”	JXFS_C_SCN_BARCODE_UPCA_E2	UPC-A/E with 2 digit add-on barcode type.
“upca_e5”	JXFS_C_SCN_BARCODE_UPCA_E5	UPC-A/E with 5 digit add-on barcode type.
“ean8_132”	JXFS_C_SCN_BARCODE_EAN8_132	EAN-8/13 with 2 digit add-on barcode type.
“ean8_135”	JXFS_C_SCN_BARCODE_EAN8_135	EAN-8/13 with 5 digit add-on barcode type.
“ean128”	JXFS_C_SCN_BARCODE_EAN128	EAN 128 barcode type.
“code93”	JXFS_C_SCN_BARCODE_CODE93	CODE 93 barcode type.
“code11”	JXFS_C_SCN_BARCODE_CODE11	CODE 11 (USD-8)
“msiplessey”	JXFS_C_SCN_BARCODE_MSIPLESSEY	MSI / PLESSEY barcode type.
“std2of5”	JXFS_C_SCN_BARCODE_STD2OF5	STANDARD 2 of 5 barcode type.
“ind2of5”	JXFS_C_SCN_BARCODE_IND2OF5	INDUSTRIAL 2 of 5 barcode type.
“code49”	JXFS_C_SCN_BARCODE_CODE49	CODE 49 barcode type.
“postnet”	JXFS_C_SCN_BARCODE_POSTNET	POSTNET barcode type.
“pdf417”	JXFS_C_SCN_BARCODE_PDF417	PDF-417 barcode type.
“datamatrix”	JXFS_C_SCN_BARCODE_DATAMATRIX	DATAMATRIX barcode type.
“maxicode”	JXFS_C_SCN_BARCODE_MAXICODE	MAXICODE barcode type.
“codeone”	JXFS_C_SCN_BARCODE_CODEONE	CODE ONE barcode type.
“channelcode”	JXFS_C_SCN_BARCODE_CHANNELCODE	CHANNEL CODE barcode type.

10 Enum Classes

All enumerations are defined in terms of a class. The following table describes all enumerated classes.

Enumeration	**Description**
JxfsScnAcquireImageEnum	possible image media-side acquisitions
JxfsScnAcquireMethodEnum	how data in a cheque scanner should be acquired
JxfsScnAdditionalProcessingSupportEnum	indicates if the device can perform additionaly processing of the media
JxfsScnAutoFeedOnEnum	whether the automatic feed is enabled or not
JxfsScnAutoFeedKindEnum	possible autoFeed capabilities a device can expose
JxfsScnAutoPresentEnum	possible autoPresent capabilities a device can expose
JxfsScnAutoSortEnum	whether the automatic sorting is enabled or not
JxfsScnBitDepthEnum	possible bit depths for acquiring images.
JxfsScnBrightnessControlEnum	specifies if brightness can be controlled by the application
JxfsScnColourModeEnum	possible modes for acquiring images
JxfsScnEncoderStatusEnum	current general status of the encoder module, if supported by the device
JxfsScnEncoderSupportEnum	whether the device supports an encoder module or not.
JxfsScnEndorserStatusEnum	current general status of the endorser module, if supported by the device
JxfsScnEndorserSupportEnum	represents the support of an endorser module and what sides of the media can be endorsed
JxfsScnEscrowStatusEnum	current status of the escrow module if supported by the device
JxfsScnEscrowSupportEnum	whether the device supports internal escrow where media can be archived after/before scanning and/or processing
JxfsScnFilterAvailableEnum	specifies if hardware can apply a red/green/blue filter over the scanning image
JxfsScnFixedHeadEnum	capability of motion of the print head of the endorsing module
JxfsScnFrontImageCaptureConfigurableEnum	whether the device can be told to capture the front image or not
JxfsScnGammaControlEnum	specifies if gamma can be controlled by the application
JxfsScnImageCaptureEnum	possible options for image capturing in a device
JxfsScnImageTypeEnum	format returned by the device service to the application when scanning images
JxfsScnItemsStatusEnum	possible states of the items at the internal escrow
JxfsScnLengthUnitEnum	possible units used to express length and position values
JxfsScnMechDesignEnum	mechanical design for a given position
JxfsScnMicrFeatureEnum	information about the MICR feature
JxfsScnOcrFeatureEnum	information about the OCR feature
JxfsScnPocketHardwareCountSupportEnum	if device is able to keep the count of the media stored in pockets or not
JxfsScnPocketStatusEnum	current status of a pocket in the module, if supported by the device
JxfsScnPocketSupportEnum	indicates if device supports handling of one or more pockets where media can be archived after scanning and/or processing
JxfsScnPositionProcessingProblemsEnum	represents an indication of any problems that may be affecting a given position.
JxfsScnContentsStatusEnum	the content status of the a position
JxfsScnRearImageCaptureConfigurableEnum	whether the device can be told to capture the rear image or not
JxfsScnResultStoredPositionEnum	posible destinations of one item after been scanned or processed
JxfsScnRetractAreaEnum	possible retract areas on the device

JxfsScnRetractSupportedEnum	if the device is able to retract the media presented to the user or not.
JxfsScnScanDuringProcessingSupportedEnum	possible behaviours of the *process()* and *processBundle()* methods in regard to simultaneous data acquiring while processing
JxfsScnScanOrderEnum	the order that has to be followed between the image acquisition and the additional process
JxfsScnScanProgressSupportEnum	if the device will fire *JXFS_I_SCAN_PROGRESS* Intermediate Events while the acquiring process is taking place
JxfsScnSharpnessControlEnum	specifies if sharpness can be controlled by the application
JxfsScnShutterCmdEnum	if explicit shutter handling is required or not.
JxfsScnStampModuleStatusEnum	represents the status of the stamp module
JxfsScnStampSupportEnum	represents the support of a stamp module in the device and what sides of the media can be stamped
JxfsScnStatusSelectorEnum	selectors to retrieve all the available status objects. For more details see status selectors chapter in *CWA Part 1:Base Architecture*.
JxfsScnStyleEnum	possible style values for an endorser module
JxfsScnTransportStatusEnum	current status of the transport module if supported by the device
JxfsScnWordWrappingSupportEnum	indicates if the endorser module will perform a word wrapping when endorsing data to the media
JxfsScnMicrDataAvailableEnum	indicates the MICR data read result

10.1 JxfsScnAcquireImageEnum

Extends	Implements
JxfsEnum	

Field	Description
front	The front side will be scanned
rear	The rear side will be scanned
frontAndRear	Both the front and the rear will be scanned

10.2 JxfsScnAcquireMethodEnum

Extends	Implements
JxfsEnum	

Field	Description
micr	MICR scanning should be used.
ocr	OCR scanning should be used.
ocrMicr	OCR and MICR scanning should be used.
unknown	The way of acquiring data is unknown

10.3 JxfsScnAdditionalProcessingSupportEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	The device supports additional processing
notSupported	The device doesn't support additional processing
unknown	It is not known whether the device supports additional processing or not.

10.4 JxfsScnAutoFeedOnEnum

Extends	Implements
JxfsEnum	

Field	Description
enabled	Auto feed is enabled.
notEnabled	Auto feed is not enabled.
unknown	It is not known whether auto feed is enabled or not.

10.5 JxfsScnAutoFeedKindEnum

Extends	Implements
JxfsEnum	

Field	Description
unknown	Auto feed capabilities are unknown.
notSupported	Auto feed capabilities are not supported in this device. The *IJxfsScnCommonControl.autoFeedOn* property is *notEnabled a*nd any call to *IJxfsScnCommonControl.setAutoFeedOn()* will throw a JXFS_E_NOT_SUPPORTED exception.
configurable	The auto feed capabilities of the device can be activated or deactivated using the *IJxfsScnCommonControl.autoFeedOn* property.
always	The auto feed capabilities of the device are always activated. The *IJxfsScnCommonControl.autoFeedOn* property is *enabled a*nd any call to *IJxfsScnCommonControl.setAutoFeedOn()* will throw a JXFS_E_NOT_SUPPORTED exception.

10.6 JxfsScnAutoPresentEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	Auto present is supported.
notSupported	Auto present is not supported.
unknown	It is not known whether auto present is supported or not.

10.7 JxfsScnAutoSortEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	Automatic sorting is enabled.
notSupported	Automatic sorting is not enabled.
unknown	It is not known whether auto sort is enabled or not.

10.8 JxfsScnBitDepthEnum

Extends	Implements
JxfsEnum	

Field	Description
lineart	1 bit/pixel black and white scanning mode.
bpp4	4 bit/pixel scanning mode (16 colors or shades of gray).
bpp8	8 bit/pixel scanning mode (256 colors or shades of gray).
bpp16	16 bit/pixel scanning mode (65,536 colors or shades of gray).
bpp24	24 bit/pixel RGB scanning mode (8 bits/color).
bpp32	32 bit/pixel RGB scanning mode (9-10 bits/color)
bpp48	48 bit/pixel RGB scanning mode (11-16 bits/color).

10.9 JxfsScnBrightnessControlEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	Brightness can be controlled by the application.
notSupported	Brightness cannot be controlled by the applicationn
unknown	It is not known whether brightness can be controlled by the application or not.

10.10 JxfsScnColourModeEnum

Extends	Implements
JxfsEnum	

Field	Description
colour	Colour scanning mode.
grayScale	Grayscale scanning mode.
blackAndWhite	Black and white scanning mode.

10.11 JxfsScnEncoderStatusEnum

Extends	Implements
JxfsEnum	

Field	Description
ok	Encoder module is in good state
inoperative	Encoder module is inoperative
unknown	Encoder module state is unknown
notSupported	Device does not support this module

10.12 JxfsScnEncoderSupportEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	The device supports an encoder module.
notSupported	The device doesn’t support an encoder module.
unknown	It is not known whether the device supports an encoder module or not.

10.13 JxfsScnEndorserStatusEnum

Extends	Implements
JxfsEnum	

Field	Description
ok	Endorser module is in good state
inoperative	Endorser module is inoperative
unknown	Endorser module state is unknown
notSupported	Device does not support this module

10.14 JxfsScnEndorserSupportEnum

Extends	Implements
JxfsEnum	

Field	Description
unknown	Endorser module availability is unknown.
notSupported	No endorser available.
front	Endorser module available for front side.
rear	Endorser module available for rear side.
both	Endorser module for both sides.

10.15 JxfsScnEscrowStatusEnum

Extends	Implements
JxfsEnum	

Field	Description
inoperative	A problem has occurred which cannot be fixed without operator intervention.
jammed	One or more items are jammed in the module. This state is usually resolved by the device service internally, however, there are situations in which automatic recovery is not possible. In these situations the JXFS_S_SCN_RESET_REQUIRED Status Event will be sent allowing the application to decide when to attempt to clear the jam. Failure to clear the jam will result in the state being escalated to '*inoperative*'.
unknown	The state of the module is unknown.
notSupported	Device does not support this module type.
ok	The escrow is at good state.

10.16 JxfsScnEscrowSupportEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	The device supports internal escrow.
notSupported	The internal escrow is not supported
unknown	It is not known whether the device supports internal escrow or not.

10.17 JxfsScnFilterAvailableEnum

Extends	Implements
JxfsEnum	

Field	Description
red	Red filter can be applied
green	Green filter can be applied.
blue	Blue filter can be applied.
redGreen	Red and/or green filters can be applied
redBlue	Red and/or blue filters can be applied
greenBlue	Green and/or blue filters can be applied
redGreenBlue	Red, green and/or blue filters can be applied.
notSupported	The device can't apply any kind of filter.
unknown	It is not known whether the device can apply a filter or not.

10.18 JxfsScnFixedHeadEnum

Extends	Implements
JxfsEnum	

Field	Description
completelyFixed	The print head of the endorsing module is completely fixed so the text will always begin at a fixed position.
partiallyFixed	The print head of the endorsing module is partially fixed so at least the Y position when the text may start could be specified.
notFixed	The print head of the endorsing module is not fixed
notSupported	The endorsing is not supported.
unknown	It is not known the capability of motion of the print head.

10.19 JxfsScnFrontImageCaptureConfigurableEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	The device can be told to capture the front image
notSupported	The device cannot be told to capture the front image.
unknown	It is not known whether the device can be told to capture the front image or not.

10.20 JxfsScnGammaControlEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	Gamma can be controlled by the application.
notSupported	Gamma cannot be controlled by the application.
unknown	It is not known whether Gamma can be controlled by the application or not.

10.21 JxfsScnImageCaptureEnum

Extends	Implements
JxfsEnum	

Field	Description
unknown	Image capture option is unknown.
notSupported	Image capture is not supported.
front	Front image capture is supported.
rear	Rear image capture is supported.
both	Both sides of media are supported for captures.

10.22 JxfsScnImageTypeEnum

Extends	Implements
JxfsEnum	

Field	Description
jpeg	JPEG Format returned.
tiff	Tiff format returned.
bmp	Windows BMP format returned.
png	Portable Network Graphics format returned.
pict	Macintosh PICT format returned.
spiff	Still Picture Interchange file format returned.
xbm	X Bitmap format returned.
pcx	PCX format returned.
vendor	Vendor Dependant format returned.
noData	The image was not requested or is not available.

10.23 JxfsScnItemsStatusEnum

Extends	Implements
JxfsEnum	

Field	Description
noItems	The escrow is empty.
itemsPresent	Items are present in the escrow.
notSupported	The hardware does not support delivering this information.
unknown	The items status is unknown.

10.24 JxfsScnLengthUnitEnum

Extends	Implements
JxfsEnum	

Field	Description
mm	The unit used by device service for length and position values is millimeter.
inch	The unit used by device service for length and position values is inch.
pixel	The unit used by device service for length and position values is pixel.
unknown	The unit used by device service for length and position values is unknown.

10.25 JxfsScnMechDesignEnum

Extends	Implements
JxfsEnum	

Field	Description
slot	This position is based on a slot design.
tray	This position is based on a tray design.

10.26 JxfsScnMicrFeatureEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	MICR reading is supported.
notSupported	MICR reading is not supported.
unknown	Is is not known whether MICR reading is supported or not.

10.27 JxfsScnOcrFeatureEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	OCR reading is supported.
notSupported	OCR reading is not supported
unknown	Is is not known whether OCR reading is supported or not.

10.28 JxfsScnPocketHardwareCountSupportEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	The device is able to keep the count of the media stored in pockets.
notSupported	The device is not able to keep the count of the media stored in pockets.
unknown	It is not known if device is able to keep the count of the media stored in pockets or not.

10.29 JxfsScnPocketStatusEnum

Extends	Implements
JxfsEnum	

Field	Description
ok	Pocket is in good state
inoperative	Pocket is inoperative
unknown	Pocket state is unknown
notSupported	Pocket status cannot be reported by the device

10.30 JxfsScnPocketSupportEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	The device supports media handling of one or more pockets.
notSupported	The device doesn’t support media handling of one or more pockets.
unknown	It is not known whether the device supports media handling or not of one or more pockets.

10.31 JxfsScnPositionProcessingProblemsEnum

Extends	Implements
JxfsEnum	

Field	Description
none	There are no problems with the position and it's associated items known.
unknown	Due to a hardware error or other condition, the state of the position cannot be determined.
metallicObjectPresent	The position contains a metallic object.
foreignObjectPresent	The position contains a foreign object.
tooManyItems	The bunch of items in the position exceeds the capacity of the position and therefore cannot be processed.
mechanicalTrouble	The items at the position cannot be processed because of mechanical problems like jammed items or a bundle wrapped with banderole.
wrongOrientation	Items are inserted, but with a wrong orientation. Cheque acceptors are usually working either short side first or long side first. Depending on the geometry of the position they may be even entered in a 90 degrees angle where they cannot be processed.

10.32 JxfsScnContentsStatusEnum

Extends	Implements
JxfsEnum	

Field	Description
empty	The position is empty.
notEmpty	The position is not empty.
unknown	The current contents in the position are unknown.
notSupported	The device cannot know if there are any contents in the position.

10.33 JxfsScnRearImageCaptureConfigurableEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	The device can be told to capture the rear image
notSupported	The device cannot be told to capture the rear image.
unknown	It is not known whether the device can be told to capture the rear image or not.

10.34 JxfsScnResultStoredPositionEnum

Extends	Implements
JxfsEnum	

Field	Description
pocket	Item is in a pocket.
escrow	Item is on the escrow.
output	Item is at the output position.
reject	Item is at the reject position.

10.35 JxfsScnRetractAreaEnum

Extends	Implements
JxfsEnum	

Field	Description
escrow	Items may be or have been retracted to the escrow.
pocket	Items may be or have been retracted to a pocket.
transport	Items may be or have been retracted to the transport.
unknown	It is not posible to know the supported retract areas.

10.36 JxfsScnRetractSupportedEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	The device is able to retract the media presented to the user.
notSupported	The device is not able to retract the media presented to the user.
unknown	It is not known if the device is able to retract the media presented to the user or not.

10.37 JxfsScnScanDuringProcessingSupportedEnum

Extends	Implements
JxfsEnum	

Field	Description
notSupported	The *process()* or *processBundle()* methods can not acquire image data. The *scan()* method must be used in order to acquire data.
alwaysBefore	The *process()* or *processBundle()* methods will always acquire image data before process the item. No call to *scan()* method is necessary if you want to perform both acquire and process operations.
alwaysAfter	The *process()* or *processBundle()* methods will always acquire image data after process the item. No call to *scan()* method is necessary if you want to perform both acquire and process operations.
optionalBefore	The *process()* or *processBundle()* methods can acquire image data, before process the item, depending on *JxfsScnProcessData,* input parameter.
optionalAfter	The *process()* or *processBundle()* methods can acquire image data, after process the item, depending on *JxfsScnProcessData,* input parameter.
unknown	It is not known yet the possible behaviour of the *process()* or *processBundle()* methods in regard to simultaneous image acquiring.

10.38 JxfsScnScanOrderEnum

Extends	Implements
JxfsEnum	

Field	Description
noScan	No image has to be aquired.
scanBeforeProcess	The image has to be aquired before the additional process.
scanAfterProcess	The image has to be aquired after the additional process.
scanProcessScan	Two images have to be aquired before and after the additional process.

10.39 JxfsScnScanProgressSupportEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	The device will fire *JXFS_I_SCAN_PROGRESS* Intermediate Events.
notSupported	The device won't fire *JXFS_I_SCAN_PROGRESS* Intermediate Events.
unknown	It is not known whether the device will fire JXFS_I_SCAN_PROGRESS Intermediate Event or not.

10.40 JxfsScnSharpnessControlEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	Sharpness can be controlled by the application.
notSupported	Sharpness cannot be controlled by the applicationn
unknown	Is not known whether sharpness can be controlled by the application or not.

10.41 JxfsScnShutterCmdEnum

Extends	Implements
JxfsEnum	

Field	Description
required	Explicit shutter handling is required.
notRequired	Explicit shutter handling is not required.
unknown	It is not known if explicit shutter handling is required or not.

10.42 JxfsScnStampModuleStatusEnum

Extends	Implements
JxfsEnum	

Field	Description
ok	The stamp module is ok
inoperative	The stamp module is not operative.
unknown	The stamp module status is unknown.
notSupported	The stamp module status can not be reported by the device.

10.43 JxfsScnStampSupportEnum

Extends	Implements
JxfsEnum	

Field	Description
front	Front stamping supported.
rear	Rear stamping supported.
both	Front and rear stamping supported.
unknown	It is not known if the stamp is supported or not.
notSupported	Stamping is not supported.

10.44 JxfsScnStatusSelectorEnum

This enumeration class is used for the base getStatus(java.util.List) method.

Extends	Implements
JxfsStatusSelectorEnum	

Field	Returned Type	Description
status	*JxfsStatus*	General status of the device.
stampInkStatus	*JxfsThresholdStatus*	Status of the ink in the scanner stamp module.
endorserInkStatus	*JxfsThresholdStatus*	Status of the ink in the scanner endorser module.
encoderInkStatus	*JxfsThresholdStatus*	Status of the ink in the scanner encoder module.
endorserStatus	*JxfsScnEndorserStatusEnum*	Generic status of the scanner endorser module.
encoderStatus	*JxfsScnEncoderStatusEnum*	Generic status of the scanner encoder module.
stampModuleStatus	*JxfsScnStampModuleStatusEnum*	Generic status of the scanner stamp module.
mediaCountersStatus	*JxfsScnMediaCounters*	Media counters status within the scanner device.
escrowStatus	*JxfsScnEscrowStatus*	Status information of the internal escrow module.
pocketStatus	*java.util.List* of *JxfsScnPocketStatus*	Status information of the supported pockets.
resetStatus	*JxfsScnResetStatus*	Provides information about the consequences of calling reset() method
positionStatus	*java.util.List* of *JxfsScnPositionStatus*	Status of the position as well as the status of the modules related with each position (transport, shutter,…,etc)
maintenanceRequired	*Boolean*	Specifies if the device is in a operational state that can't be recovered without the intervention of an operator or not.
autoFeedOn	*JxfsScnAutoFeedOnEnum*	Status of the automatic document feeder
transportStatus	*JxfsScnTransportStatusEnum*	Status of the transport unit.

10.45 JxfsScnStyleEnum

Extends	Implements
JxfsEnum	

Field	Description
standard	Standard style.
bold	Bold style.
compressed	Compressed style.
underline	Underline style.
italics	Italics style.

10.46 JxfsScnTransportStatusEnum

Extends	Implements
JxfsEnum	

Field	Description
ok	Module is in a good state.
inoperative	A problem has occurred which cannot be fixed without operator intervention.
jammed	One or more items are jammed in the module. This state is usually resolved by the device service internally, however, there are situations in which automatic recovery is not possible. In these situations the JXFS_S_SCN_RESET_REQUIRED Status Event will be sent allowing the application to decide when to attempt to clear the jam. Failure to clear the jam will result in the state being escalated to '*inoperative*'.
unknown	The state of the module is unknown.
notSupported	The device does not support this module.

10.47 JxfsScnWordWrappingSupportEnum

Extends	Implements
JxfsEnum	

Field	Description
supported	The endorser module will perform a word wrapping
notSupported	The endorser module won't perform a word wrapping
unknown	It is not known whether the endorser module will perform a word wrappping or not

10.48 JxfsScnMicrDataAvailableEnum

Extends	Implements
JxfsEnum	

Field	Description
notFound	MICR data was not found.
found	MICR data was found and read.
notRead	MICR data was found but can't be read.

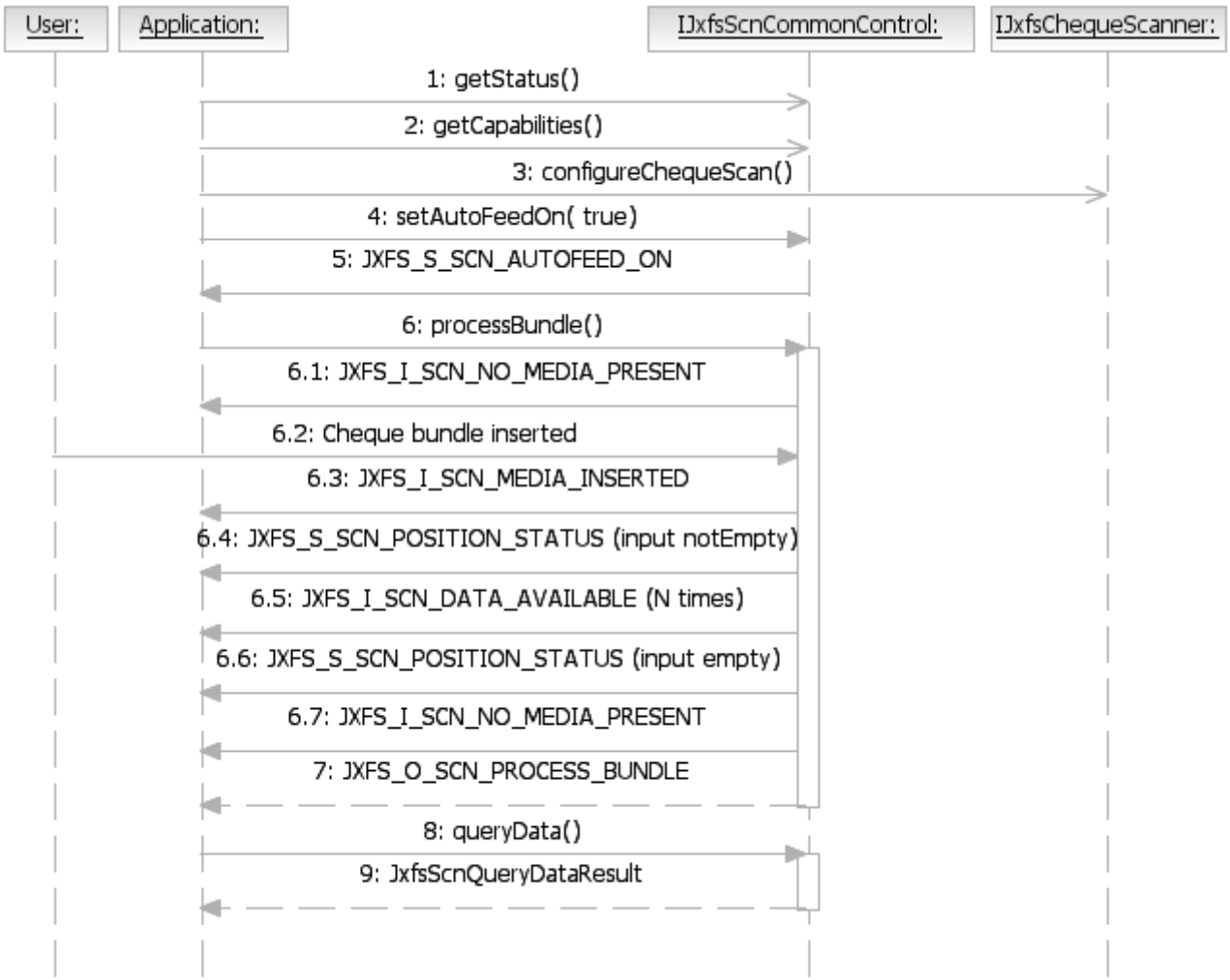
11 Sequence Diagrams

This section provides some sequence diagrams of typical situations managing scanner devices.

11.1 Simple cheque bundle process with autoFeed

In this example we will use a desktop Cheque Scanner with two output pockets, one for valid cheques and the other one for the rejected ones. The device will automatically select the output pocket based on data readed from each media. The list below shows some of the device capabilities (JxfsScnCapabilities):

- autoFeed = configurable
- escrowSupported = *notSupported*
- pocketsSupported = *supported*
- defaultInputPosition = JXFS_C_SCN_POS_TOP
- positionsCapabilities[0]
  - position = JXFS_C_SCN_POS_TOP
  - contentsStatusSupported = true
  - input = true



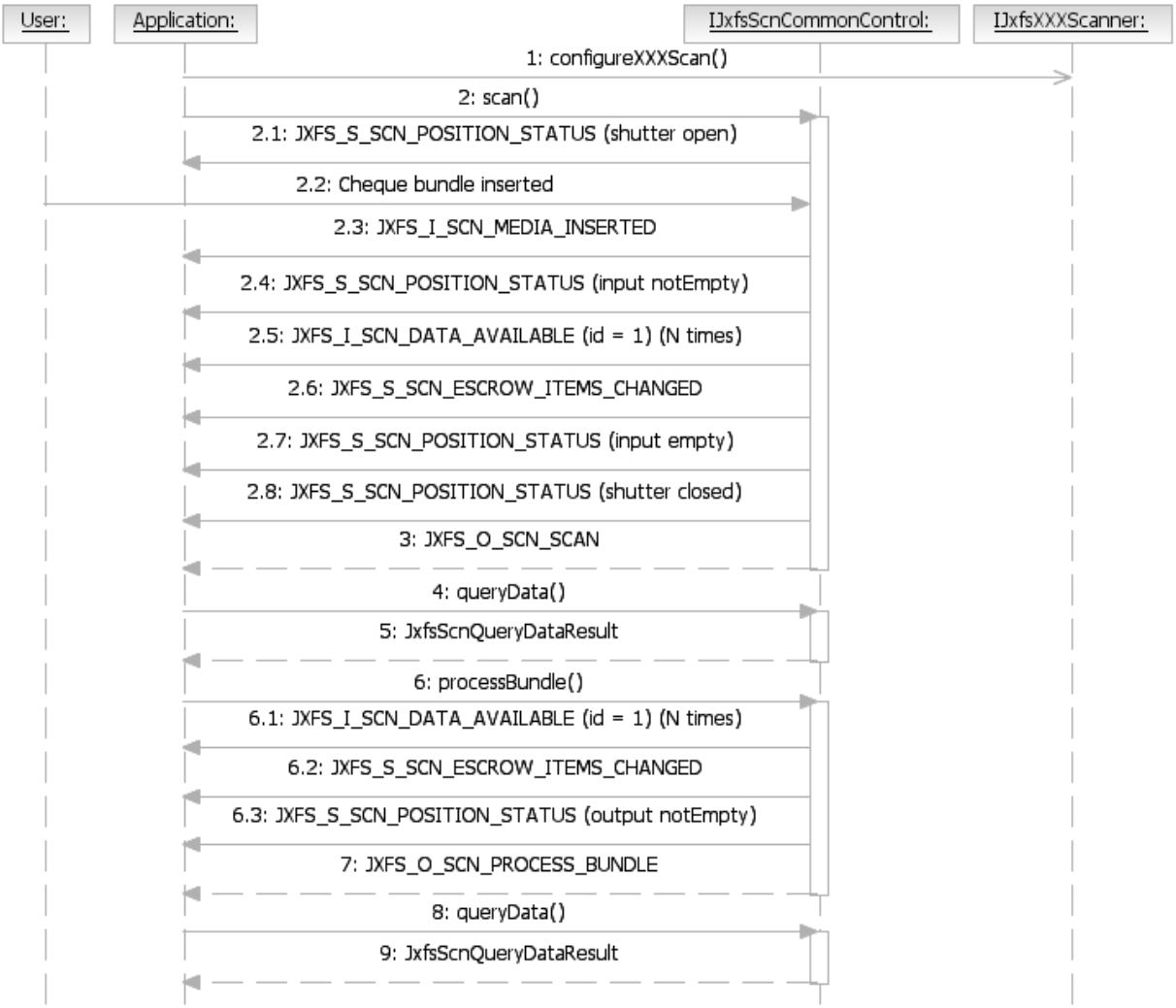
11.2 Scan - process - queryData with autoFeed.

In this example it is going to be assumed the following premises:

- The device supports shutter but does not need to use specific commands to control it.
- The device also supports pockets. To be precise, it supports 2 pockets
- The device supports auto feed.
- The device supports internal escrow.

The resulting JxfsScnCapabilities object will be:

- autoFeed = configurable
- escrowSupported = *supported*
- pocketsSupported = *supported*
- shutterCmd = *notRequired*
- defaultInputPosition = JXFS_C_SCN_POS_TOP
- positionsCapabilities[0]
    - position = JXFS_C_SCN_POS_TOP
    - shutterStatusSupported = true
    - shutterCmd = false
    - contentsStatusSupported = true
    - input = true



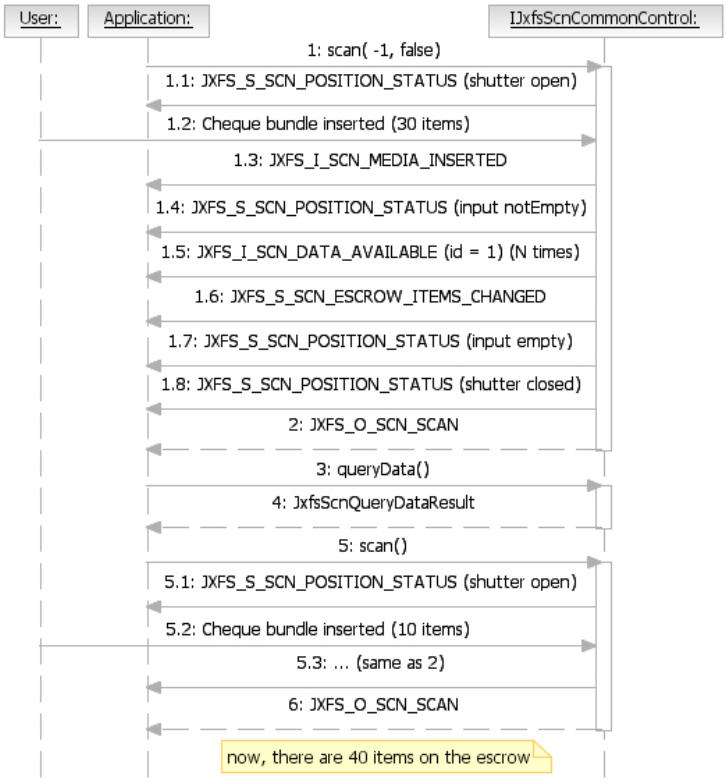
Steps 8 and 9 are optional depending on device capabilities and customer requirements.

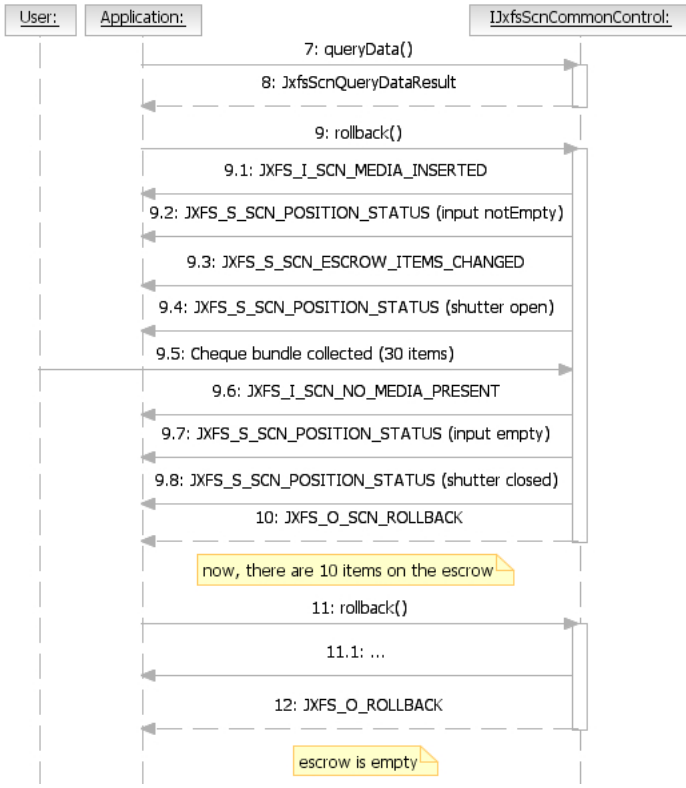
11.3 Multiple bundle handling from escrow to output position

Expanding the diagram of 11.2 section we will show how to perform a *rollback* when the escrow has more items than the output position supports. After two consecutive bundle scans from the input position to the escrow, the application must execute *rollback* commands until the escrow is empty. After the first rollback the amount of items present in the escrow can be derived from the *rollback* OCE data (containing a *JxfsScnRollbackResult* object) or by checking the JxfsScnStatus.escrowStatus.contents.count.

The JxfsScnCapabilities object from section 11.2 will be expanded with:

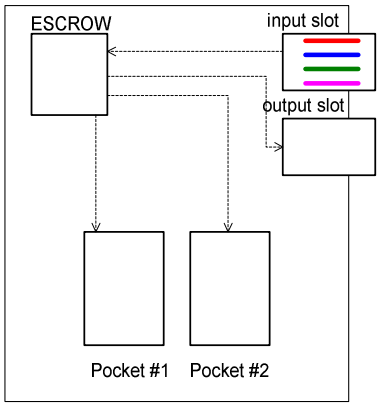
- maxItemsPerBundle = 50
- defaultOutputPosition = JXFS_C_SCN_POS_BOTTOM
- positionsCapabilities[1]
  - position = JXFS_C_SCN_POS_BOTTOM
  - maxItems = 30
  - output = true

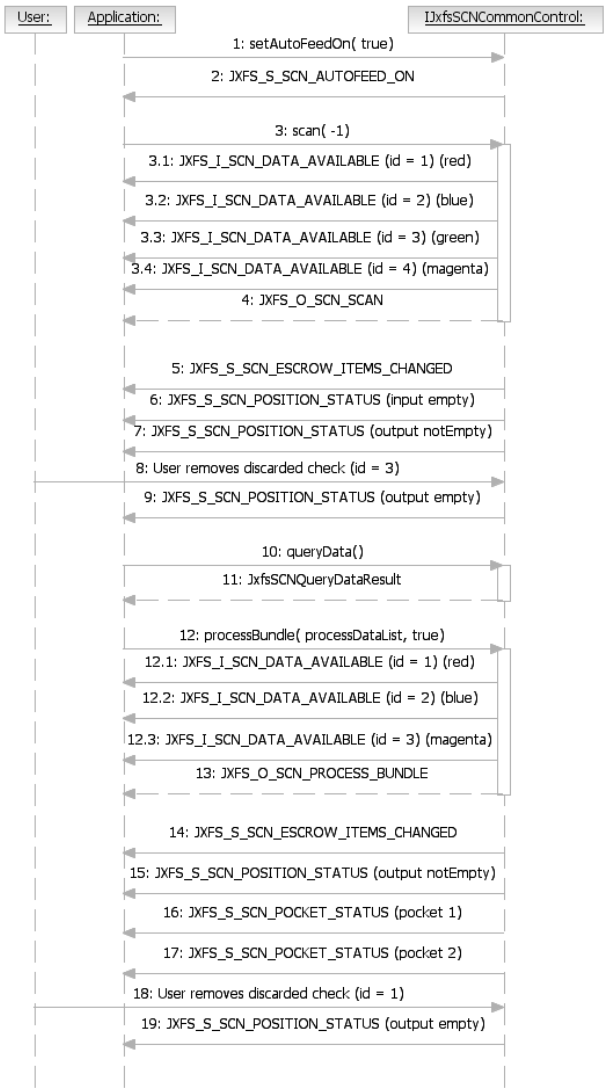




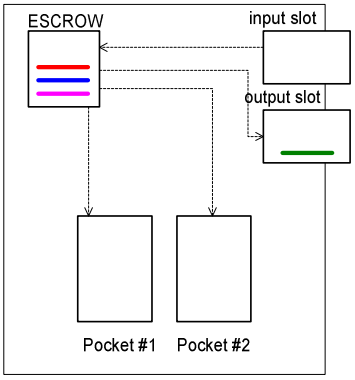
11.4 Complex Bundle Cheque Handling: Distribution to several pockets

Given is an input bundle with 4 cheques.

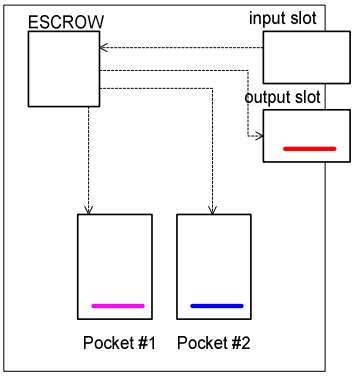




When scanning these cheques, one of them is discarded (green). Possible reasons are wrong formats, bad MICR, etc. Usually these cheques have to be removed before continuing (similar to cash-in).



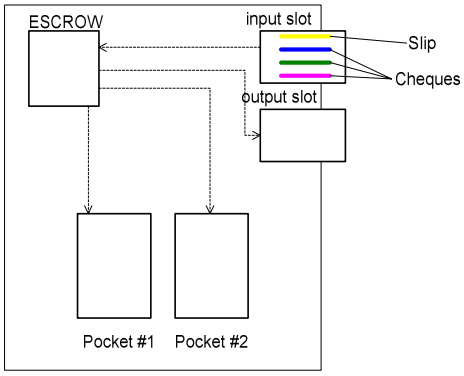
With the three remaining cheques on the ESCROW, the application will process them in a way that one will be put in Pocket#1 (magenta), one in Pocket#2 (blue) and one will be returned (red). Possible reasons for returning them is that the scanned result does not fit the requirements (cheque from wrong bank, etc.).

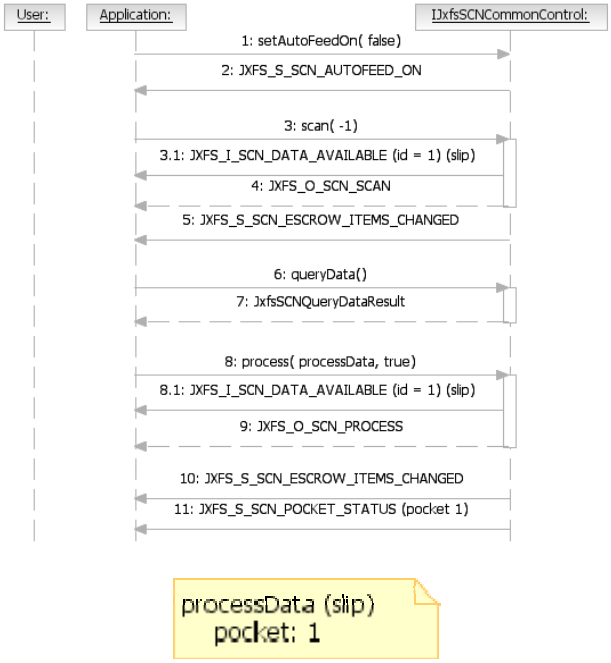



```
processDataList:
  item 1 (red)
    pocket : JXFS_C_SCN_VALUE_NOT_INITIALIZED
    position : output
  item 2 (blue)
    pocket : 2
  item 3 (magenta)
    pocket : 1
```

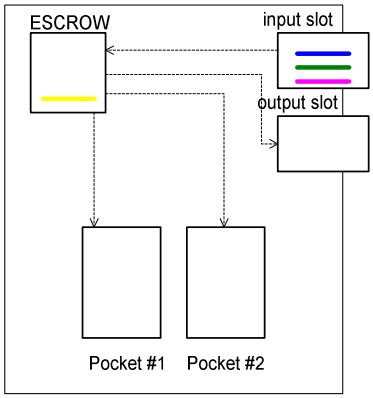
11.5 Complex Bundle Cheque Handling: Usage of Slips

Primary in the UK Market the usage of a slip is common. So a standard use case for a slip is the following.

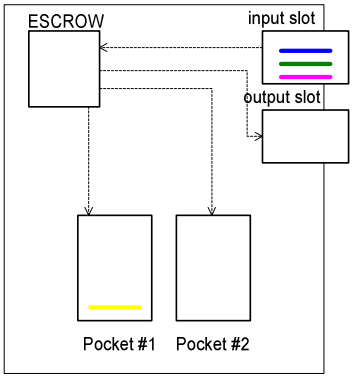


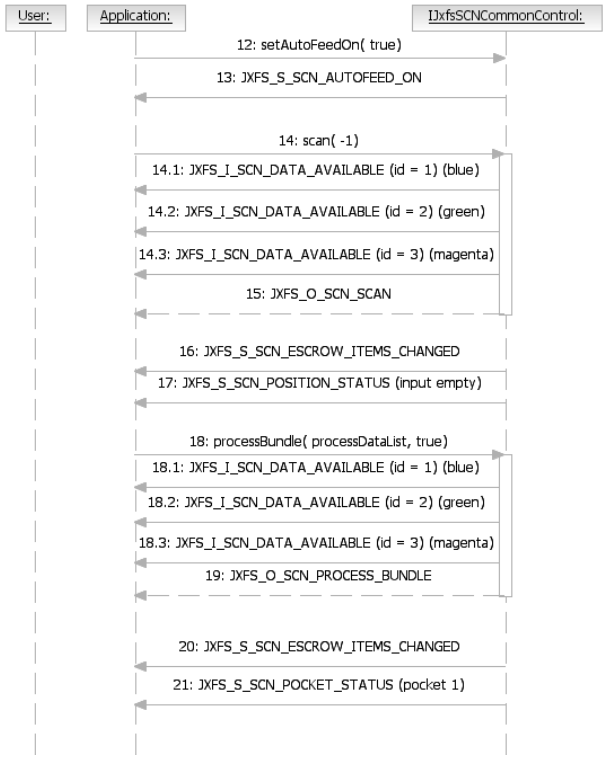


The bundle of cheques will be entered with the slip on top of it. The device then scans the slip only.

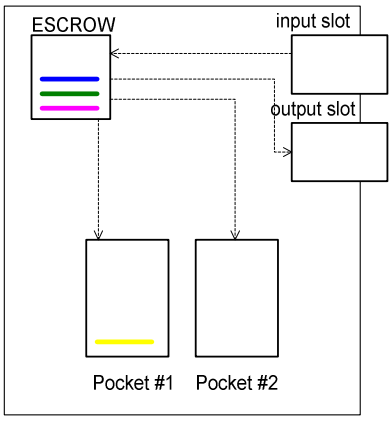


If the result of the scanning of the slip is successfull, it will be deposited into a pocket, otherwise returned.



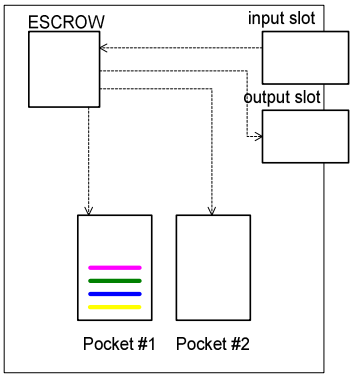


Then the rest of the bundle will be scanned.



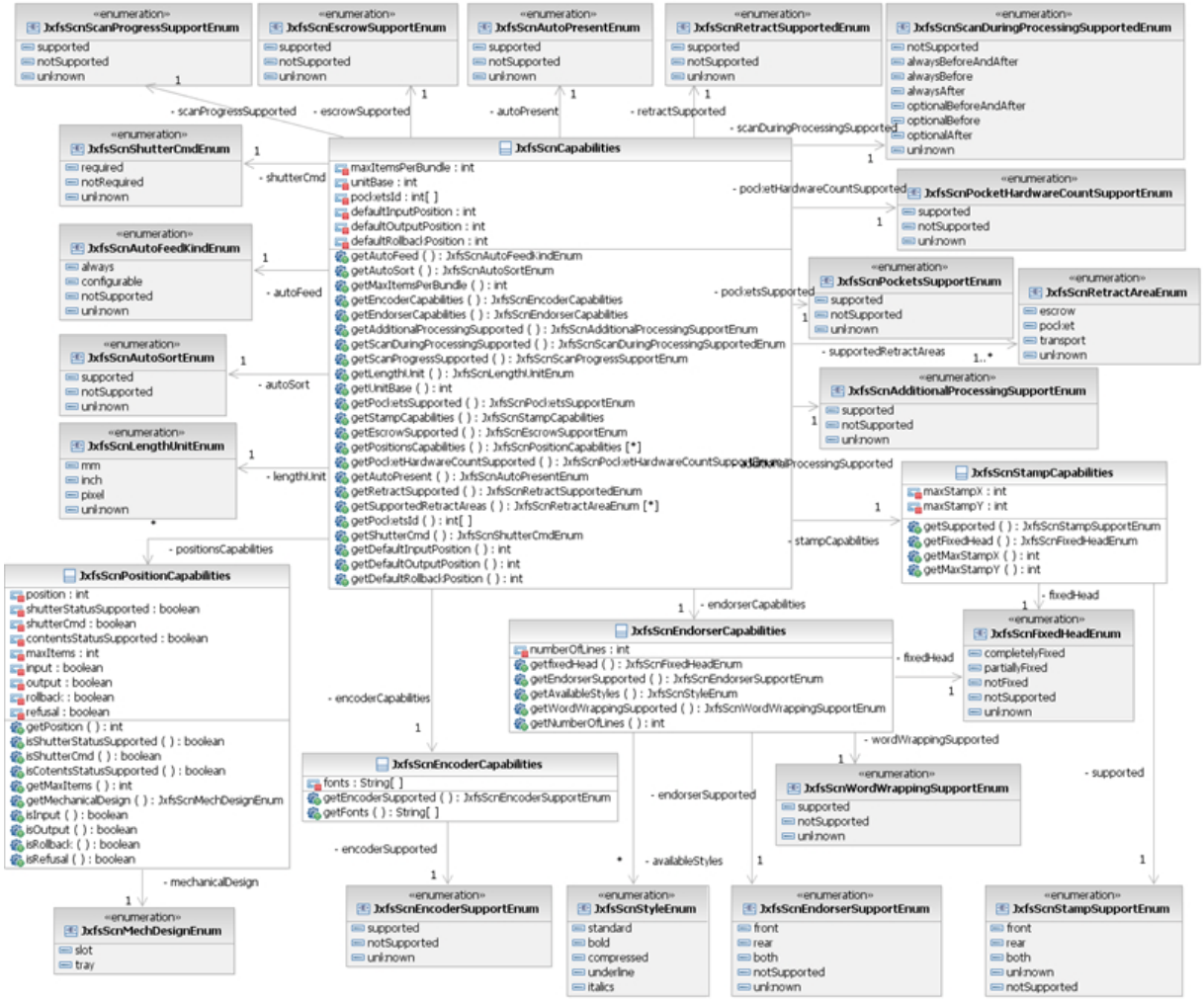

```
processDataList:
  item 1 (blue)
    pocket : 1
  item 2 (green)
    pocket : 1
  item 3 (magenta)
    pocket : 1
```

If all of the scanned cheques are OK, they are usually put into the same pocket, separated from other users cheques by slips.

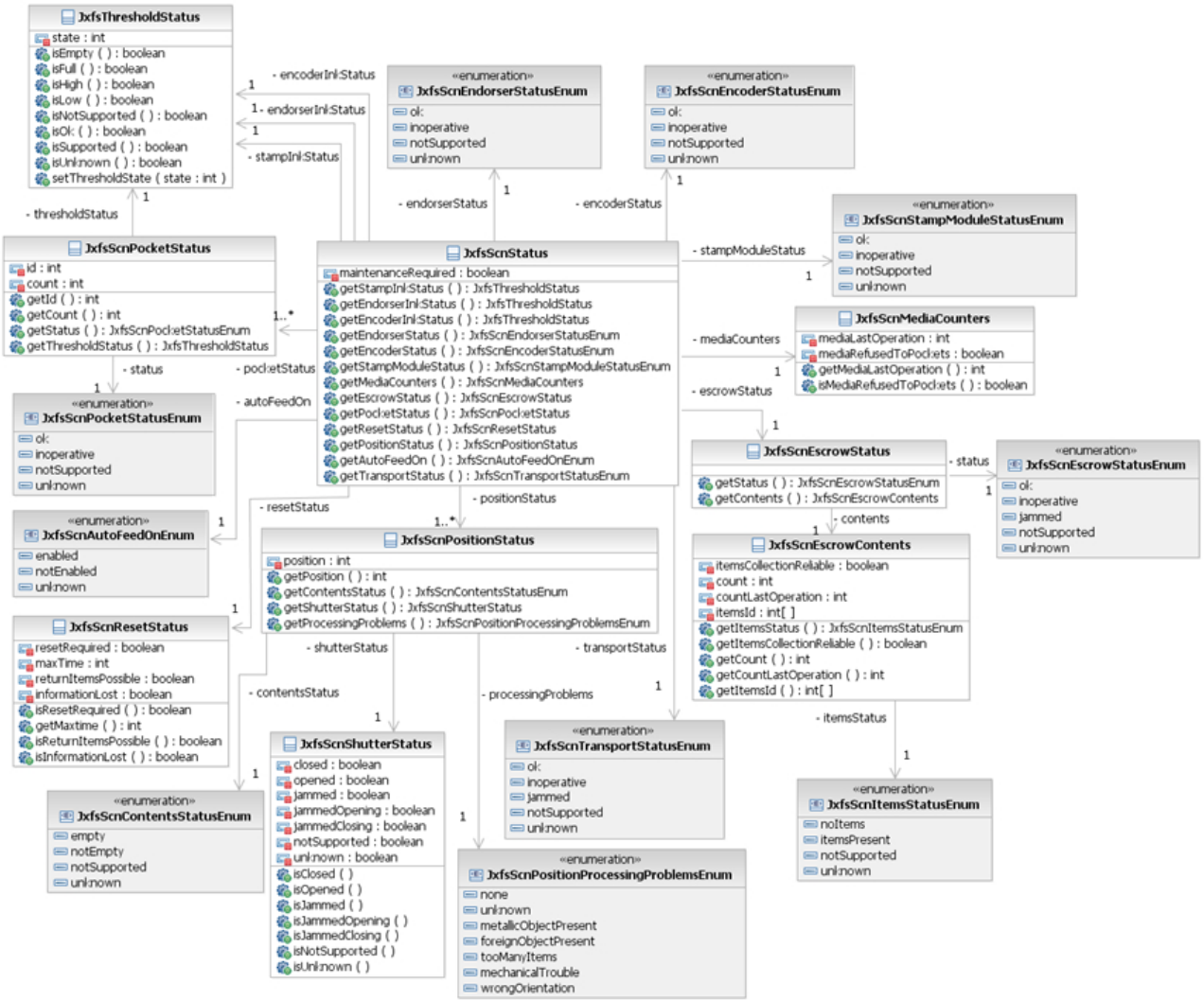


12 Extended Class Diagrams

12.1 JxfsScnCapabilities



12.2 JxfsScnStatus



12.3 JxfsScnResult