# CEN

# WORKSHOP

# AGREEMENT

# CWA 14923-6

May 2004

**ICS** 35.240.40

Supersedes CWA 13937-6:2003

English version

# J/eXtensions for Financial Sevices (J/XFS) for the Java Platform - Part 6: Printer Device Class Interface - Programmer's Reference



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

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



Ref. No.:CWA 14923-6:2004 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/ISSS J/XFS Workshop. J/XFS provides an API for Java applications which need to access financial devices. It is hardware independent and, by using 100% pure Java, also operating system independent.

The CEN/ISSS J/XFS Workshop gathers suppliers (among others the J/XFS Forum members), service providers as well as banks and other financial service companies. A list of companies participating in this Workshop and in support of this CWA is available from the CEN/ISSS Secretariat. The specification was agreed upon by the J/XFS Workshop Meeting of 2002-09-25/26 in Barcelona and a subsequent electronic review by the Workshop participants, and the final version was sent to CEN for publication on 2002-12-06.

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

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

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

This CWA is composed of the following parts:

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

CWA 14923-6:2004 replaces CWA 13937-6:2003 and should be read in conjunction with CWA 13937-6:2000, which contains the previous release of the J/XFS specification

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

# History

Major changes from CWA 13937-62000 document

- *IJxfsRetract* interface extends the *IJxfsEject* interface. According to this, the *JxfsPassbookPrinter* and *JxfsDocumentPrinter* classes don't implement the *IJxfsEject* interface directly.
- Added Clarifications considering handling of *null* parameter values.
- Definitions of terms "paper", "media" and "exit/entry slot" added.
- General error code JXFS_E_FAILURE added.
- General error codes may also be reported as results in operation completion events.
- Property *statusCapability* added to the *IJxfsPrinterControl* interface.
- Status event with the code JXFS_S_PTR_DEVICE removed.
- All *OCPtr** classes were removed. The *JxfsOperationCompleteEvent* class with appropriate operation codes and data objects are used instead.
- Error codes added.:
  - JXFS_E_PTR_MEDIA_JAM
  - JXFS_E_PTR_TONER_EMPTY
  - JXFS_E_PTR_EXIT_ENTRY_FAILURE
  - JXFS_E_PTR_INK_EMPTY
  - JXFS_E_PTR_STACKER_FULL
- Status codes added:
  - JXFS_S_PTR_EXIT_ENTRY
  - JXFS_S_PTR_STACKER
  - JXFS_S_PTR_STACKERCOUNT
- The JXFS_E_PTR_FIELD_FAILURE constant replaced with JXFS_I_PTR_FIELD_FAILURE.
- The method *getFieldDescription* of the *IJxfsPrinterControl* interface returns data about all fields if *null* is passed as *fieldNames* parameter.
- Indices in the *printForm* method of the *IJxfsPrinterControl* interface are enclosed in square brackets ('[', ']').
- Lists of possible error codes and status events for *printRawData* and *reset* methods of the *IJxfsPrinterControl* interface were significantly changed.
- New properties in the *IJxfsEject* interface: *ejectStatusCapability*, *exitEntryStatus*, *stackerCount* and *stackerStatus*.
- The *inkStatus* property was removed from the *IJxfsRetrack* interface because it is already contained in *IJxfsEject*.
- The property *readStatusCapability* added to the *IJxfsRead* interface.
- The *readForm* method with 3 parameters added to the *IJxfsRead* interface. The *readForm* method with 1 parameter was marked as deprecated.
- The *readImage* method with 3 parameters added to the *IJxfsRead* interface. The *readImage* method with 1 parameter was marked as deprecated.
- Support classes added:
  - *JxfsPtrEjectStatusCapability*
  - *JxfsPtrReadStatusCapability*
  - *JxfsPtrStackerCount*
  - *JxfsPtrStatusCapability*
- The properties *formsDescriptionList* and *mediaDescriptionList* of the *JxfsPtrFormsConfig* class marked as deprecated.
- Status classes added:
  - *JxfsPtrExitEntryStatus*

The method *isLampNotSupported* of the *JxfsPtrLampStatus* class marked as deprecated

# 1 Scope

This document describes the printer device class based on the basic architecture of J/XFS which is similar to the JavaPOS architecture. It is event driven and asynchronous.

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

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

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

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

# 2 Overview

## 2.1 Description

The J/XFS Printer Device Support allows for the operation of the following categories of printers:

- **Receipt Printer**
  The receipt printer is used to print cut sheet documents. It may or may not require insert or eject operations, and often includes an operator identification device, e.g., Teller A and Teller B lights, for shared operation.

- **Journal Printer**
  The journal is a continuous form device used to record a hardcopy audit trail of transactions, and for certain report printing requirements.

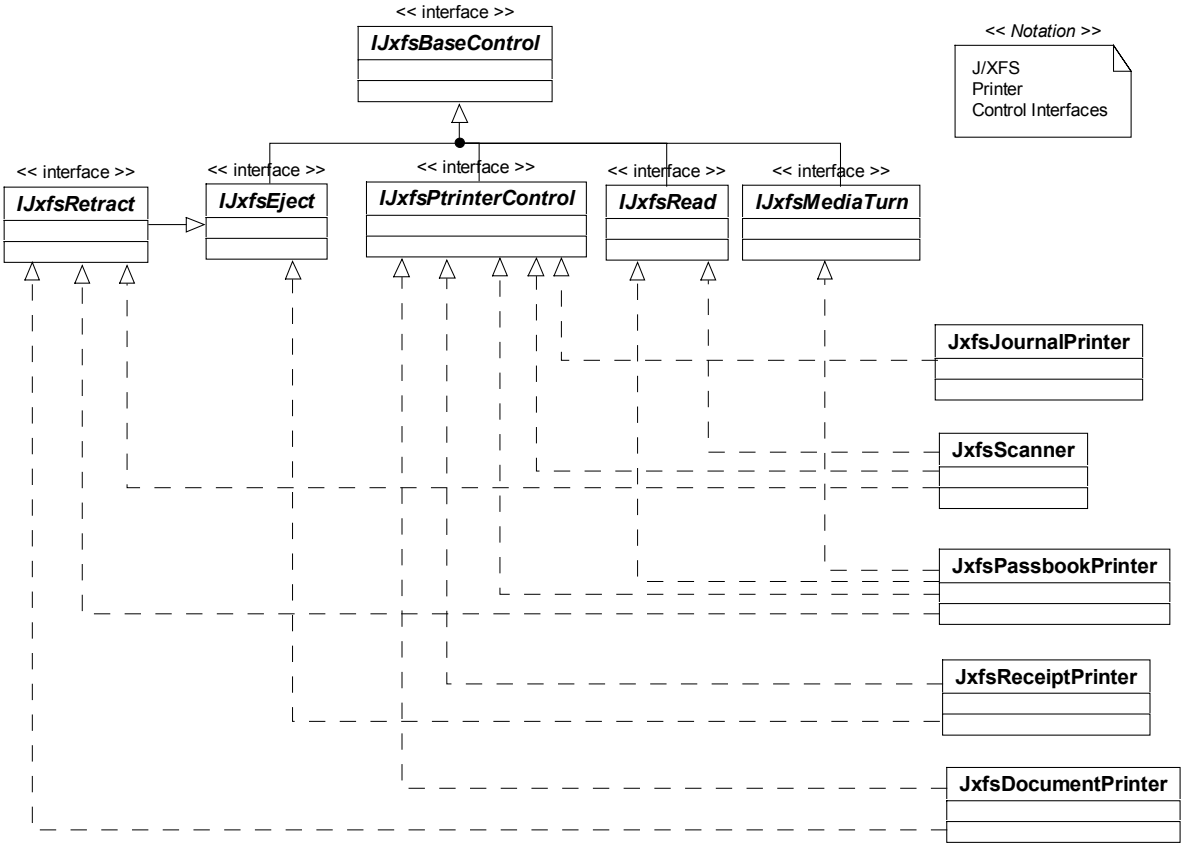
- **Passbook Printer**
  The passbook device is physically and functionally the most complex printer. The J/XFS definition supports automatic positioning of the book, as well as read/write capability for an optional integrated magnetic stripe. The implementation also manages the geometry of the book - i.e. the margins and centerfolds - presenting the simplest possible application interface while delivering the full range of functionality.

- **Document Printer**
  Document printing is similar to receipt printing -- a set of fields are positioned on an inserted sheet of paper -- but the focus is on full-size forms. It should be noted that the J/XFS environment only implements the printing of text fields from the application. The electronic printing of the form image itself is not supported; but can be delivered as an added-value extension by the vendor. Statement printers belong to this category

- **Scanner**
  The scanner device is able to scan any inserted printed or handwritten media. It may also be capable of printing.
  The J/XFS definition supports automatic positioning of the inserted media, as well as read/write capability.

The J/XFS Printer Device Support uses the event driven model. The application will instantiate a J/XFS Printer Device Control Object and then call the defined I/O methods with passing data objects containing the parameters. When an I/O method is called, the J/XFS Printer Device Support will attempt to process the requested I/O. If the request is invalid or an exception is encountered the application will be notified by a J/XFS exception. Completion of the request will be reported by an event. Thus the application must register itself with the J/XFS Printer Device Control Object for the various types of events it wishes to handle. If forms are being used then the J/XFS Printer Device Service will access the form indicated by the application via the published J/XFS configuration interface and use the form data to define positioning and presentation information for each of the fields on the document.

## 2.2 Class Hierarchy



## 2.3 Class and Interface Summary

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

| Class or Inter-face | Name | Description | Extends / Implements |
|---|---|---|---|
| Inter-face | **IJxfsBaseControl** | Base interface for all device controls. Contains methods specific to all the device controls. | -- |
| Class | **JxfsBaseControl** | Base class for all device controls. Implements the methods defined in the **IJxfsBaseControl** Interface. Contains the properties specific to all device controls. | Implements: **IJxfsBaseControl** |
| Inter-face | **IJxfsPrinterControl** | Base interface for all printer controls. Contains the methods specific to all the device controls for the printer device category. | Extends: **IJxfsBaseControl** |
| Inter-face | **IJxfsEject** | Interface that contains methods for the eject functionality of receipt printers, passbook printers, document printers and scanners. | Extends: **IJxfsBaseControl** |
| Inter-face | **IJxfsMediaTurn** | Interface that contains methods to turn media inside a printer | Extends: **IJxfsBaseControl** |
| Inter-face | **IJxfsRetract** | Interface that contains methods for the retract functionality of passbook printers, document printers and scanners. | Extends: **IJxfsEject** |
| Inter-face | **IJxfsRead** | Interface that contains methods for the read functionality of scanners and passbook printers. | Extends: **IJxfsBaseControl** |
| Class | **JxfsDocumentPrinter** | Class for the Document Printer control | Implements: **IJxfsPrinterControl** **IJxfsRetract** |
| Class | **JxfsJournalPrinter** | Class for the Journal Printer control | Implements: **IJxfsPrinterControl** |

| Class or Inter-face | Name | Description | Extends / Implements |
|---|---|---|---|
| Class | **JxfsPassbookPrinter** | Class for the Passbook Printer control. | Implements: **IJxfsPrinterControl IJxfsRetract IJxfsMediaTurn IJxfsRead** |
| Class | **JxfsReceiptPrinter** | Class for the Receipt Printer control. | Implements: **IJxfsPrinterControl IJxfsEject** |
| Class | **JxfsScanner** | Class for the Scanner control. | Implements: **IJxfsPrinterControl IJxfsRetract IJxfsRead** |
| Inter-face | **IJxfsEventNotification** | Includes one callback method per event type. The Device Service calls these methods to cause events to be delivered to the application. | -- |

# 3 Device behavior

## 3.1 Base service behavior

The basic printer device behavior conforms to the CWA specification, Part 1: J/eXtensions for Financial Services (J/XFS) for the Java Platform - Base Architecture - Programmer's Reference.

## 3.2 Handling of *null* parameters

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

## 3.3 Paper vs. media

The specification refers to the terms paper and media. When the term paper is used this refers to paper that is situated in a paper supply attached to the printer. The term media is used for media that is inserted by the customer (e.g. check and other material that is scanned) or that is issued to the customer (e.g. a receipt or statement). That means that a journal printer has only paper and scanners have only media. Receipt, document printers and also passbook printers with white passbook dispensing capability have both. As soon as the paper is in the print position it becomes media.

## 3.4 Exit / entry slot

The term “exit / entry slot” refers to the physical position within the printer device where the inserting of the media by the customer occurs (e.g. check, passbook and other material that is scanned, read or written by the device) or where the customer takes the media ejected by the device. The *IJxfsEject* interface defines methods and events for handling the states of the exit / entry slot if a printer device has the capability to determine it.

# 4 Classes and Interfaces

All operation methods return an identificationID. If a method cannot be processed immediately a JxfsException is thrown. After processing has taken place, an *OperationCompleteEvent* is generated which contains detailed information about the status of the operation, i.e. if it failed or succeeded, and eventually additional data as a result.

Used support classes, status classes and constants are described in additional chapters.

## 4.1 Access to properties

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

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

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

### get*Property*

| | |
|---|---|
| **Syntax** | ***Property getProperty(void) throws JxfsException;*** |
| **Description** | Returns the requested property value. |
| **Parameter** | None |
| **Event** | No additional events are generated. |
| **Exceptions** | See section on JxfsExceptions for all JxfsException value codes. Some possible JxfsException codes are.<br>JXFS_E_CLOSED<br>JXFS_E_REMOTE<br>JXFS_E_UNREGISTERED |

### set*Property*

| | |
|---|---|
| **Syntax** | ***void setProperty(Property) throws JxfsException;*** |
| **Description** | Sets the requested property. |
| **Parameter** | Single parameter of the ***Property*** type, representing the new property value. |
| **Event** | No additional events are generated. |
| **Exceptions** | See section on JxfsExceptions for all JxfsException value codes. Some possible JxfsException codes are.<br>JXFS_E_CLOSED<br>JXFS_E_PARAMETER_INVALID<br>JXFS_E_REMOTE<br>JXFS_E_UNREGISTERED |

## 4.2 Exceptions

The methods described for the specific interfaces can all throw a *JxfsException*. The exception error codes which can be thrown in all methods are described in the table below:

| Error Code | Meaning |
|---|---|
| JXFS_E_CLOSED | The Device Control is closed. Use *open()* first. |
| JXFS_E_PARAMETER_INVALID | At least one method argument has an invalid value. |
| JXFS_E_NOT_SUPPORTED | The method is (currently) not supported. |
| JXFS_E_REMOTE | An error happened in the communication layer. |
| JXFS_E_UNREGISTERED | The Device Control is not registered. |
| JXFS_E_FAILURE | A general error code for an unclassified failure within an operation. |

Those error code can also appear as the *result* value within *OperationCompleteEvent* events. Only if a method can throw an exception with an additional error code or send an

*OperationCompleteEvent* event with a different result, it is explicitly mentioned in this document.

## 4.3 IJxfsPrinterControl

The J/XFS Printer Device Control Subclass is defined in *JxfsPrinterControl* and is a subclass of *JxfsBaseControl*. Its interface is defined in *IJxfsPrinterControl* which extends the *IJxfsBaseControl* interface. The intent of the J/XFS Printer Device Control object is to allow data and control to pass between the application and the device support code so that the associated device can be accessed.

The various status events are sent whenever the state of the underlying physical device changes, independently of the execution of the defined operations.

### 4.3.1 Summary

| Property | Type | Access |
|---|---|---|
| compound | **boolean** | R |
| ctrlMediaCapability | **JxfsPtrCtrlMediaCapability** | R |
| statusCapability | **JxfsPtrStatusCapability** | R |
| extentCapability | **JxfsPtrExtentCapability** | R |
| formsConfig | **JxfsPtrFormsConfig** | R/W |
| ptrStatus | **JxfsPtrStatus** | R |
| writeFormCapability | **JxfsPtrWriteFormCapability** | R |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| set*Property* | void |
| is*Property* | boolean |
| ctrlMedia | identificationID |
| getFormList | identificationID |
| mediaExtents | identificationID |
| getMediaList | identificationID |
| printForm | identificationID |
| printRawData | identificationID |
| getFieldDescription | identificationID |
| getFormDescription | identificationID |
| getMediaDescription | identificationID |
| resetPrinter | identificationID |

| Event | May occur during / after |
|---|---|
| StatusEvent | |
| JXFS_S_PTR_MEDIA | *ctrlMedia(), mediaExtents(), printForm(), printRawData(), resetPrinter()* |
| JXFS_S_PTR_PAPER | *printForm(), printRawData(), resetPrinter()* |
| JXFS_S_PTR_TONER | *printForm(), printRawData(), resetPrinter()* |
| IntermediateEvent | |
| JXFS_I_PTR_ NO_MEDIA_PRESENT | *printForm(), printRawData(), mediaExtents()* |
| JXFS_I_PTR_MEDIA_INSERTED | *printForm(), printRawData(), mediaExtents()* |
| JXFS_I_PTR_FIELD_FAILURE | *printForm()* |

| Event | May occur during / after |
|---|---|
| OperationCompleteEvent | |
| JXFS_O_PTR_CTRL_MEDIA | *ctrlMedia()* |
| JXFS_O_PTR_FIELD_INFO | *getFieldDescription()* |
| JXFS_O_PTR_FORM_INFO | *getFormDescription()* |
| JXFS_O_PTR_FORM_LIST | *getFormList()* |
| JXFS_O_PTR_MEDIA_INFO | *getMediaDescription()* |
| JXFS_O_PTR_MEDIA_LIST | *getMediaList()* |
| JXFS_O_PTR_MEDIA_EXTENTS | *mediaExtent()* |
| JXFS_O_PTR_WRITE_FORM_DATA | *printForm()* |
| JXFS_O_PTR_WRITE_RAW_DATA | *printRawData()* |
| JXFS_O_PTR_RESET_PRINTER | *resetPrinter()* |

### 4.3.2 Properties

#### compound (R)

This property is deprecated. It is mentioned here for compatibility reasons only. It's value has no practical meaning and should be ignored. The query method *isCompound()* is also deprecated.

#### ctrlMediaCapability (R)

**Type** ***JxfsPtrCtrlMediaCapability***
**Initial Value** see *JxfsPtrCtrlMediaCapability*
**Description** This property defines capabilities for special handling of the print media.

#### extentCapability (R)

**Type** ***JxfsPtrExtentCapability***
**Initial Value** see *JxfsPtrExtentCapability*
**Description** This property defines printer capabilities for measuring the media extents.

#### statusCapability (R)

**Type** ***JxfsPtrStatusCapability***
**Initial Value** see *JxfsPtrStatusCapability*
**Description** This property defines printer capabilities for determining states of it's components.

#### formsConfig (R/W)

**Type** ***JxfsPtrFormsConfig***
**Initial Value** see *JxfsPtrFormsConfig*
**Description** This property defines the general forms configuration.

#### ptrStatus (R)

**Type** ***JxfsPtrStatus***
**Initial Value** see *JxfsPtrStatus*
**Description** This property encapsulates the state of the printer device. Every printer status change is reported by the Device Service. The Device Control sends the corresponding *StatusEvent* to all registered listeners.

#### writeFormCapability (R)

**Type** ***JxfsPtrWriteFormCapability***
**Initial Value** see *JxfsWriteFormCapability*
**Description** This property specifies printer capabilities to write forms.

### 4.3.3 Methods

Please note that forms, fields and media names can be any valid strings. They are matched case sensitively.

### ctrlMedia

**Syntax** ***identificationID ctrlMedia(int mediaControl) throws JxfsException;***

**Description** This command is used to control a form drawn in by the device (e.g. after reading or in case of termination of an application request).

**Parameter**

| Type | Name | Meaning |
|---|---|---|
| *int* | mediaControl | Specifies the manner in which the media should be handled, as a combination of the following flags :<br>JXFS_PTR_CTRL_ALARM<br>JXFS_PTR_CTRL_FLUSH<br>JXFS_PTR_CTRL_SKIP<br>For descriptions of those flags see chapter 7: "Constants". |

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated :

**OperationCompleteEvent**

When a *ctrlMedia()* operation is completed an *OperationCompleteEvent* will be sent by J/XFS Printer Device Control to all registered listeners with the following data:

| Field | Value |
|---|---|
| *operationID* | JXFS_O_PTR_CTRL_MEDIA |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_FLUSH_FAIL<br>JXFS_E_PTR_MEDIA_JAM<br>JXFS_E_PTR_NO_MEDIA_PRESENT |
| *data* | none |

**StatusEvent**

When the status of the media changes a *StatusEvent* is sent to all registered listeners with the following data :

| Field | Value |
|---|---|
| *status* | JXFS_S_PTR_MEDIA |
| *details* | JxfsMediaStatus mediaStatus<br>The new printer media status. |

### getFieldDescription

**Syntax** ***identificationID getFieldDescription(String[] fieldNames, String formName) throws JxfsException;***

**Description** This method is used to retrieve details of the definition of a single or all fields on a specified form. *fieldNames* and *formName* will be used to define fields whose definitions are requested.

**Parameter**

| Type | Name | Meaning |
|---|---|---|
| String[] | fieldNames | Names of the requested fields. If this parameter is *null* then descriptions of all fields are returned, otherwise descriptions of only those fields named are returned.<br>The array is not allowed to contain *null* entries. |
| String | formName | Name of the requested form. |

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated :

**OperationCompleteEvent**

When a *getFieldDescription()* operation is completed an *OperationCompleteEvent* event will be sent by J/XFS Printer Device Control to the registered listeners with the following data:

| Field | Value |
|---|---|
| *operationID* | JXFS_O_PTR_FIELD_INFO |
| *identificationID* | The corresponding ID |

| | | |
|---|---|---|
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_FIELD_NOT_FOUND<br>JXFS_E_PTR_FORM_NOT_FOUND | |
| *data* | JxfsPtrField[] aFieldDefs<br>List of the field definitions available on the specified form. If an error occurs this field will be *null* as no field descriptions could be returned. | |

### getFormDescription

**Syntax** ***identificationID getFormDescription(String formName) throws JxfsException;***

**Description** This method is used to retrieve details of the definition of a specified form. *formName* will be used to define the form whose definition is requested.

| **Parameter** | **Type** | **Name** | **Meaning** |
|---|---|---|---|
| | String | formName | Name of the requested form. |

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated:

**OperationCompleteEvent**

When a *getFormDescription()* operation is completed an *OperationCompleteEvent* event will be sent by J/XFS Printer Device Control to the registered listeners with the following data:

| **Field** | **Value** |
|---|---|
| *operationID* | JXFS_O_PTR_FORM_INFO |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_FORM_NOT_FOUND |
| *data* | JxfsPtrForm aJxfsPtrForm<br>Description of the requested form. If an error occurs this field will be *null* as no form description could be returned. |

### getFormList

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

**Description** This method is used to retrieve a list of the names of the form definitions available on the printer.

**Parameter** **None**

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated :

**OperationCompleteEvent**

When a *getFormList()* operation is completed an *OperationCompleteEvent* event will be sent by J/XFS Printer Device Control to the registered listeners with the following data:

| **Field** | **Value** |
|---|---|
| *operationID* | JXFS_O_PTR_FORM_LIST |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_NOFORMS |
| *data* | String[] aFormsList<br>List of the forms available on the printer. If an error occurs this field will be *null* as no form list could be returned. |

### getMediaDescription

**Syntax** ***identificationID getMediaDescription(String mediaName) throws JxfsException;***

**Description** This method is used to retrieve details of the definition of a specified media. *mediaName* will be used to define the media whose definition is desired.

| **Parameter** | **Type** | **Name** | **Meaning** |
|---|---|---|---|
| | String | mediaName | Name of the requested media. |

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated :

**OperationCompleteEvent**

When a *getMediaDescription()* operation is completed an *OperationCompleteEvent* event will be sent by J/XFS Printer Device Control to the registered listeners with the following data:

| **Field** | **Value** |
|---|---|
| *operationID* | JXFS_O_PTR_MEDIA_INFO |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_MEDIA_NOT_FOUND |
| *data* | JxfsPtrMedia aJxfsPtrMedia<br>Description of the requested media. If an error occurs this field will be *null* as no media description could be returned. |

### getMediaList

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

**Description** This method is used to retrieve a list of names of the media definitions available on the printer.

**Parameter** **None**

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated :

**OperationCompleteEvent**

When a *getMediaList()* operation is completed an *OperationCompleteEvent* event will be sent by J/XFS Printer Device Control to the registered listeners with the following data:

| **Field** | **Value** |
|---|---|
| *operationID* | JXFS_O_PTR_MEDIA_LIST |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_NOMEDIA |
| *data* | String[] aMediaList<br>List of media definitions available on the printer. If an error occurs this field will be *null* as no media list could be returned. |

### mediaExtents

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

**Description** This method is used to get the extents of the media inserted in the printer. The extents will be based on the values of *formsConfig.base, formsConfig.unitX* and *formsConfig.unitY*. If no media is present the printer waits endlessly for media to be inserted, or until cancelled by the application.

**Parameter** **None**

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated :

**OperationCompleteEvent**

When a *mediaExtents()* operation is completed an *OperationCompleteEvent* event will be sent by J/XFS Printer Device Control to the registered listeners with the following data:

| **Field** | **Value** |
|---|---|
| *operationID* | JXFS_O_PTR_MEDIA_EXTENTS |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_MEDIA_JAM |
| *data* | *JxfsPtrMediaExtents* aJxfsPtrMediaExtents<br>The extents of the inserted media. If an error occurs this field will be *null* as no media extents could be returned. |

**IntermediateEvent**
If no media is present the J/XFS Printer Device Control will send an *IntermediateEvent* to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *operationID* | JXFS_O_PTR_MEDIA_EXTENTS |
| *identificationID* | The corresponding ID |
| *reason* | JXFS_E_PTR_NO_MEDIA_PRESENT |
| *data* | none |

**IntermediateEvent**
If media is inserted and the operation can continue the J/XFS Printer Device Control will send an *IntermediateEvent* to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *operationID* | JXFS_O_PTR_MEDIA_EXTENTS |
| *identificationID* | The corresponding ID |
| *reason* | JXFS_I_PTR_MEDIA_INSERTED |
| *data* | none |

**StatusEvent**
When the status of the media changes a *StatusEvent* is sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_MEDIA |
| *details* | JxfsMediaStatus mediaStatus<br>The new printer media status. |

### printForm

**Syntax** ***identificationID printForm(String formName, String mediaName, String[] fieldWriteData) throws JxfsException;***

**Description** This method prints the form with the name *formName* using the description of the media defined by *mediaName.* After a successful completion of this output operation, an *OperationCompleteEvent* is issued to inform the application of the results. If no media is present the printer waits endlessly for media to be inserted, or until cancelled by the application.

Printers with paper source (e.g. journal and receipt printers) will send an *OperationCompleteEvent* with the JXFS_E_PTR_PAPEROUT result if they run out of paper during printing. The application should be aware that some printing might still have occurred.

**Parameter**

| **Type** | **Name** | **Meaning** |
|---|---|---|
| String | formName | Name of the form to be printed. |
| String | mediaName | Name of the media to be used for printing. |
| String[] | fieldWriteData | An array of "<FieldName>=<FieldValue>" strings. If the field is an index field, then the syntax of the field is instead "<FieldName>[<index>]=<FieldValue>" where <index> indicates the zero based element of the index field. For example, the string "Street[5]=Unknown" denotes the 6th element of the indexed field with the name "Street" should be printed with the value "Unknown".<br>This array is not allowed to contain *null* entries. |

**Exceptions** No additional exceptions generated.
**Events** Following events can be generated :

**OperationCompleteEvent**
When a *printForm()* operation is completed an *OperationCompleteEvent* will be sent by J/XFS Printer Device Control to all registered listeners with the status containing the following data:

| | |
|---|---|
| *operationID* | JXFS_O_PTR_WRITE_FORM_DATA |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_FIELD_ERROR<br>JXFS_E_PTR_FIELD_NOT_FOUND<br>JXFS_E_PTR_FIELD_SPEC_FAILURE<br>JXFS_E_PTR_FORM_INVALID<br>JXFS_E_PTR_FORM_NOT_FOUND<br>JXFS_E_PTR_MEDIA_INVALID<br>JXFS_E_PTR_MEDIA_JAM<br>JXFS_E_PTR_MEDIA_NOT_FOUND<br>JXFS_E_PTR_MEDIA_OVERFLOW<br>JXFS_E_PTR_MEDIA_SKEWED<br>JXFS_E_PTR_PAPEROUT<br>JXFS_E_PTR_TONER_EMPTY |
| *data* | none |

**IntermediateEvent**
If no media is present the J/XFS Printer Device Control will send an *IntermediateEvent* to all registered listeners with the following data:

| | |
|---|---|
| *operationID* | JXFS_O_PTR_WRITE_FORM_DATA |
| *identificationID* | The corresponding ID |
| *reason* | JXFS_I_PTR_ NO_ MEDIA_PRESENT |
| *data* | none |

**IntermediateEvent**
If media is inserted and the operation can continue the J/XFS Printer Device Control will send an *IntermediateEvent* to all registered listeners with the following data :

| | |
|---|---|
| *operationID* | JXFS_O_PTR_WRITE_FORM_DATA |
| *identificationID* | The corresponding ID |
| *reason* | JXFS_I_PTR_MEDIA_INSERTED |
| *data* | none |

**IntermediateEvent**
If a field error occurs during printing the field and the Device Service is capable to continue with printing the further fields[1], an *IntermediateEvent* will be sent to all registered listeners with the following data:

| | |
|---|---|
| *operationID* | JXFS_O_PTR_WRITE_FORM_DATA |
| *identificationID* | The corresponding ID |
| *reason* | JXFS_I_PTR_FIELD_FAILURE |
| *data* | JxfsPtrFieldFailure failure<br>More detailed information about the failure. |

**StatusEvent**
When the status of the printer's paper supply changes a *StatusEvent* is sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_PAPER |
| *details* | JxfsThresholdStatus paperStatus<br>The new paper supply status. |

**StatusEvent**
When the status of the printer's toner supply changes a *StatusEvent* is

[1] An abrupt termination of the form printing may be defined by the *overflow* property of the *JxfsPtrField* object or by some device-specific conditions.

sent to all registered listeners with the following data :

| Field | Value |
|---|---|
| *status* | JXFS_S_PTR_TONER |
| *details* | JxfsThresholdStatus tonerStatus<br>The new toner supply status. |

**StatusEvent**
When the status of the media changes a *StatusEvent* is sent to all registered listeners with the following data :

| Field | Value |
|---|---|
| *status* | JXFS_S_PTR_MEDIA |
| *details* | JxfsMediaStatus media<br>The new printer media status. |

### printRawData

**Syntax** ***identificationID printRawData(byte[] rawData, boolean inputData) throws JxfsException;***

**Description** This command is used to send raw data (a byte string of device dependent data) to the physical device. If no media is present the printer waits endlessly for media to be inserted, or until cancelled by the application. If input data was expected (see parameter inputData) and was sent to the Device Service object, the *data* property of the *OperationCompleteEvent* is initialized properly.

Printers with paper source (e.g. journal and receipt printers) will send an *OperationCompleteEvent* with the JXFS_E_PTR_PAPEROUT result if they run out of paper during printing. The application should be aware that some printing might still have occurred.

The *printRawData()* method should be used with great care, because the raw data can also include some escape sequences containing printer commands which won't be recognized by the Device Service. Hence, the Device Service will not be able to correctly update it's state objects. This could cause an unpredictable behavior. For the same reason, various error codes can be returned as the *result* field of the *OperationCompleteEvent* event.

**Parameter**

| Type | Name | Meaning |
|---|---|---|
| byte[] | rawData | Raw data to be sent to the printer. |
| boolean | inputData | Indicates whether input data from the printer is expected in response to sending the raw data. This may be the case if the application uses this method to send some printer-specific commands not covered by J/XFS (e.g. loading fonts) and is interested in data returned by the printer. This flag informs the Device Service to wait for the printer response instead of returning as soon as raw data is sent. |

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated :
**OperationCompleteEvent**
When a *printRawData()* operation is completed an *OperationCompleteEvent* will be sent by J/XFS Printer Device Control to all registered listeners with the status containing the following data:

| | |
|---|---|
| *operationID* | JXFS_O_PTR_WRITE_RAW_DATA |
| *identificationID* | The corresponding ID |

| | |
|---|---|
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_EXIT_ENTRY_FAILURE<br>JXFS_E_PTR_INK_EMPTY<br>JXFS_E_PTR_MEDIA_JAM<br>JXFS_E_PTR_MEDIA_SKEWED<br>JXFS_E_PTR_PAPEROUT<br>JXFS_E_PTR_RETRACT_BIN_FULL<br>JXFS_E_PTR_STACKER_FULL<br>JXFS_E_PTR_TONER_EMPTY |
| *data* | byte[] inputData<br>Input data sent by the printer. The value is *null* if no input data was expected and/or the input data has not been sent to the Device Service object by the printer. |

**IntermediateEvent**
If no media is present the J/XFS Printer Device Control will send an *IntermediateEvent* to all registered listeners with the following data :

| | |
|---|---|
| *operationID* | JXFS_O_PTR_WRITE_RAW_DATA |
| *identificationID* | The corresponding ID |
| *reason* | JXFS_I_PTR_NO_MEDIA_PRESENT |
| *data* | none |

**IntermediateEvent**
If media is inserted and the operation can continue the J/XFS Printer Device Control will send an *IntermediateEvent* to all registered listeners with the following data :

| | |
|---|---|
| *operationID* | JXFS_O_PTR_WRITE_RAW_DATA |
| *identificationID* | The corresponding ID |
| *reason* | JXFS_I_PTR_MEDIA_INSERTED |
| *data* | none |

**StatusEvent**
Various status events are sent during this operation, whenever the status value changes.

### resetPrinter

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

**Description** Resets the printer. The Device Service should try to put the printer device in it's initial state. This may include ejecting the current printing media, but it is not obligatory. The operational state of the printer can be determined after this operation by using the *getPtrStatus()* method.

**Parameter** **None**

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated :

**OperationCompleteEvent**
When a *resetPrinter()* operation is completed an *OperationCompleteEvent* will be sent by J/XFS Printer Device Control to all registered listeners with the following data:

| | |
|---|---|
| *operationID* | JXFS_O_PTR_RESET_PRINTER |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL (or any of the general error codes described in the chapter 4.2) |
| *data* | none |

**StatusEvent**
Various status events are sent during this operation, whenever the status value changes.

## 4.4 IJxfsEject

### 4.4.1 Summary

| Property | Type | Access |
|---|---|---|
| ejectStatusCapability | **JxfsPtrEjectStatusCapability** | R |
| exitEntryStatus | **JxfsPtrExitEntryStatus** | R |
| inkStatus | **JxfsThresholdStatus** | R |
| maxStackerCapability | **JxfsPtrMaxStackerCapability** | R |
| stackerCount | **JXfsPtrStackerCount** | R/W |
| stackerStatus | **JxfsThresholdStatus** | R |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| ejectMedia | identificationID |
| prepareEject | identificationID |

| Event | May occur during / after |
|---|---|
| StatusEvent | |
| JXFS_S_PTR_EXIT_ENTRY | *ejectMedia()* |
| JXFS_S_PTR_STACKER | *ejectMedia(), prepareEject()* |
| JXFS_S_PTR_STACKERCOUNT | *ejectMedia(), prepareEject()* |
| JXFS_S_PTR_INK | *ejectMedia(), prepareEject()* |
| JXFS_S_PTR_MEDIA | *ejectMedia(), prepareEject()* |
| OperationCompleteEvent | |
| JXFS_O_PTR_EJECT_MEDIA | *ejectMedia()* |
| JXFS_O_PTR_PREPARE_EJECT | *prepareEject()* |

### 4.4.2 Properties

**ejectStatusCapability (R)**

**Type** ***JxfsPtrEjectStatusCapability***
**Initial Value** see *JxfsPtrEjectStatusCapability*
**Description** This property defines the printer's capabilities to determine the states of it's eject components.

**exitEntryStatus (R)**

**Type** ***JxfsPtrExitEntryStatus***
**Initial Value** see *JxfsThresholdStatus*
**Description** This property defines the printer's exit / entry slot status.

**inkStatus (R)**

**Type** ***JxfsThresholdStatus***
**Initial Value** see *JxfsThresholdStatus*
**Description** This property defines the stamping ink cartridge status.

**maxStackerCapability (R)**

**Type** ***JxfsPtrMaxStackerCapability***
**Initial Value** see *JxfsPtrMaxStackerCapability*
**Description** This property defines the capacity of the printer's eject stacker.

**stackerCount (R/W)**

**Type** ***JxfsStackerCount***
**Initial Value** see *JxfsStackerCount*
**Description** This property represents the number of stacked medias prior to eject.

**stackerStatus (R)**

**Type** ***JxfsThresholdStatus***

**Initial Value** see *JxfsThresholdStatus*
**Description** This property defines the printer's stacker status.

## 4.4.3 Methods

### ejectMedia

**Syntax** ***identificationID ejectMedia(int mediaControl) throws JxfsException;***
**Description** This command is used to eject a form. The operation completes as soon as the ejected media is available at the exit / entry slot of the device.

| **Parameter** | **Type** | **Name** | **Meaning** |
|---|---|---|---|
| | *int* | mediaControl | Specifies the manner in which the media should be handled before ejecting, as a combination of the following flags:<br>JXFS_PTR_CTRL_ALARM<br>JXFS_PTR_CTRL_FLUSH<br>JXFS_PTR_CTRL_SKIP<br>JXFS_PTR_CTRL_CUT<br>JXFS_PTR_CTRL_PARTIALCUT<br>JXFS_PTR_CTRL_PERFORATE<br>JXFS_PTR_CTRL_STACK<br>JXFS_PTR_CTRL_STAMP<br>For descriptions of those flags see chapter 7: "Constants". |

**Exceptions** No additional exceptions generated.
**Events** Following events can be generated :

**OperationCompleteEvent**
When an *ejectlMedia()* operation is completed an *OperationCompleteEvent* will be sent by J/XFS Printer Device Control to all registered listeners with the following data:

| **Field** | **Value** |
|---|---|
| *operationID* | JXFS_O_PTR_EJECT_MEDIA |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_EXIT_ENTRY_FAILURE<br>JXFS_E_PTR_FLUSH_FAIL<br>JXFS_E_PTR_INK_EMPTY<br>JXFS_E_PTR_MEDIA_JAM<br>JXFS_E_PTR_NO_MEDIA_PRESENT<br>JXFS_E_PTR_STACKER_FULL |
| *data* | none |

**StatusEvent**
When the status of the media changes a *StatusEvent* is sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_MEDIA |
| *details* | JxfsMediaStatus mediaStatus<br>The new printer media status. |

**StatusEvent**
When the status of the exit / entry slot changes a *StatusEvent* is sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_EXIT_ENTRY |
| *details* | JxfsPtrExitEntryStatus exitEntryStatus<br>The new printer exit slot status. |

**StatusEvent**
When the stamping ink cartridge status changes a *StatusEvent* is sent to all registered listeners with the following data :

| Field | Value |
|---|---|
| *status* | JXFS_S_PTR_INK |
| *details* | JxfsThresholdStatus inkStatus<br>The new printer stamp ink cartridge status. |

**StatusEvent**
When the status of the stacker changes a *StatusEvent* will be sent to all registered listeners with the following data :

| Field | Value |
|---|---|
| *status* | JXFS_S_PTR_STACKER |
| *details* | JxfsThresholdStatus stackerStatus<br>The new stacker status. |

**StatusEvent**
When the status of the stacker counter changes a *StatusEvent* will be sent to all registered listeners with the following data :

| Field | Value |
|---|---|
| *status* | JXFS_S_PTR_STACKERCOUNT |
| *details* | JxfsStackerCount stackerCount<br>The new stacker counter value. |

### prepareEject

**Syntax** ***identificationID prepareEject(int mediaControl) throws JxfsException;***

**Description** This command is used to prepare the ejecting of a printed form. On printers which have the ability to stack media prior to eject, the JXFS_PTR_CTRL_STACK *mediaControl* flag can be used in subsequent calls of this method in order to stack more pages and then eject them as a bundle using the *ejectMedia()* method.
The operation completes when the media is handled in the way defined by the *mediaControl* parameter. Then an *OperationCompleteEvent* is sent.

**Parameter**

| Type | Name | Meaning |
|---|---|---|
| *int* | mediaControl | Specifies the manner in which the media should be prepared for ejecting, as a combination of the following flags :<br>JXFS_PTR_CTRL_ALARM<br>JXFS_PTR_CTRL_FLUSH<br>JXFS_PTR_CTRL_SKIP<br>JXFS_PTR_CTRL_CUT<br>JXFS_PTR_CTRL_PARTIALCUT<br>JXFS_PTR_CTRL_PERFORATE<br>JXFS_PTR_CTRL_STACK<br>JXFS_PTR_CTRL_STAMP<br>For descriptions of those flags see chapter 7: "Constants". |

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated :

**OperationCompleteEvent**
When a *prepareEject()* operation is completed an *OperationCompleteEvent* will be sent by J/XFS Printer Device Control to all registered listeners with the following data:

| Field | Value |
|---|---|
| *operationID* | JXFS_O_PTR_PREPARE_EJECT |
| *identificationID* | The corresponding ID |

| | |
|---|---|
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_FLUSH_FAIL<br>JXFS_E_PTR_INK_EMPTY<br>JXFS_E_PTR_MEDIA_JAM<br>JXFS_E_PTR_NO_MEDIA_PRESENT<br>JXFS_E_PTR_STACKER_FULL |
| *data* | none |

**StatusEvent**
When the status of the media changes a *StatusEvent* is sent to all registered listeners with the following data:

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_MEDIA |
| *details* | JxfsMediaStatus mediaStatus<br>The new media status. |

**StatusEvent**
When the stamping ink cartridge status changes a *StatusEvent* is sent to all registered listeners with the following data:

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_INK |
| *details* | JxfsThresholdStatus inkStatus<br>The new printer stamping ink cartridge status. |

**StatusEvent**
When the status of the stacker changes a *StatusEvent* will be sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_STACKER |
| *details* | JxfsThresholdStatus stackerStatus<br>The new stacker status. |

**StatusEvent**
When the status of the stacker counter changes a *StatusEvent* will be sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_STACKERCOUNT |
| *details* | JxfsStackerCount stackerCount<br>The new stacker counter value. |

## 4.5 IJxfsRetract

### 4.5.1 Summary

| Property | Type | Access |
|---|---|---|
| maxRetractCapability | **JxfsPtrMaxRetractCapability** | R |
| retractBinStatus | **JxfsThresholdStatus** | R |
| retractCount | **JxfsPtrRetractCount** | R/W |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| set*Property* | void |
| retractMedia | identificationID |

| Event | May occur during / after |
|---|---|
| StatusEvent | |
| JXFS_S_PTR_EXIT_ENTRY | *retractMedia()* |
| JXFS_S_PTR_INK | *retractMedia()* |
| JXFS_S_PTR_MEDIA | *retractMedia()* |
| JXFS_S_PTR_RETRACT_BIN | *retractMedia()* |
| JXFS_S_PTR_RETRACTCOUNT | *retractMedia()* |
| OperationCompleteEvent | |
| JXFS_O_PTR_RETRACT_MEDIA | *retractMedia()* |

### 4.5.2 Properties

#### maxRetractCapability (R)

**Type** ***JxfsPtrMaxRetractCapability***
**Initial Value** see *JxfsThresholdStatus*
**Description** This property defines the capacity of the printer's retract bin.

#### retractBinStatus (R)

**Type** ***JxfsThresholdStatus***
**Initial Value** see *JxfsThresholdStatus*
**Description** This property defines the printer's retract bin status.

#### retractCount (R/W)

**Type** ***JxfsRetractCount***
**Initial Value** see *JxfsRetractCount*
**Description** This property represents the number of retracted medias.

### 4.5.3 Methods

#### retractMedia

**Syntax** ***identificationID retractMedia(int mediaControl) throws JxfsException;***
**Description** This command is used to retract a form by the device after it has been presented to the user in the entry / exit slot. The JXFS_E_NO_MEDIA_PRESENT error code is used when there is no media in the entry / exit slot of device.
**Parameter** **Type** **Name** **Meaning**

| | | | |
|---|---|---|---|
| | *int* | mediaControl | Specifies the manner in which the media should be handled before retracting, as a combination of the following flags:<br>JXFS_PTR_CTRL_ALARM<br>JXFS_PTR_CTRL_FLUSH<br>JXFS_PTR_CTRL_CUT<br>JXFS_PTR_CTRL_STAMP<br>For descriptions of those flags see chapter 7: "Constants". |

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated:

**OperationCompleteEvent**

When a *retractMedia()* operation is completed an *OperationCompleteEvent* will be sent by J/XFS Printer Device Control to all registered listeners with the following data:

| **Field** | **Value** |
|---|---|
| *operationID* | JXFS_O_PTR_RETRACT_MEDIA |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_EXIT_ENTRY_FAILURE<br>JXFS_E_PTR_INK_EMPTY<br>JXFS_E_PTR_MEDIA_JAM<br>JXFS_E_PTR_NO_MEDIA_PRESENT<br>JXFS_E_PTR_RETRACT_BIN_FULL |
| *data* | none |

**StatusEvent**

When the status of the media changes a *StatusEvent* is sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_MEDIA |
| *details* | JxfsMediaStatus mediaStatus<br>The new printer media status. |

**StatusEvent**

When the status of the retract bin changes a *StatusEvent* will be sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_RETRACT_BIN |
| *details* | JxfsThresholdStatus retractBinStatus<br>The new retract bin status. |

**StatusEvent**

When the status of the retract counter changes a *StatusEvent* will be sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_RETRACTCOUNT |
| *details* | JxfsRetractCount retractCount<br>The new retract counter value. |

**StatusEvent**

When the status of the exit slot changes a *StatusEvent* is sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_EXIT_ENTRY |
| *details* | JxfsThresholdStatus exitSlotStatus<br>The new printer exit slot status. |

**StatusEvent**

When the stamping ink cartridge status changes a *StatusEvent* is sent to all registered listeners with the following data :

| Field | Value |
|---|---|
| *status* | JXFS_S_PTR_INK |
| *details* | JxfsThresholdStatus inkStatus<br>The new printer stamp ink cartridge status. |

## 4.6 IJxfsMediaTurn

### 4.6.1 Summary

| Property | Type | Access |
|---|---|---|
| ctrlTurnCapability | **JxfsPtrCtrlTurnCapability** | R |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| atpBackward | identificationID |
| atpForward | identificationID |
| turnMedia | identificationID |

| Event | May occur during / after |
|---|---|
| StatusEvent<br>JXFS_S_PTR_MEDIA | <br>*atpBackward(), atpForward(), turnMedia()* |
| OperationCompleteEvent<br>JXFS_O_PTR_ATP_BACKWARD<br>JXFS_O_PTR_ATP_FORWARD<br>JXFS_O_PTR_TURN_MEDIA | <br>*atpBackward()*<br>*atpForward()*<br>*turnMedia()* |

### 4.6.2 Properties

#### ctrlTurnCapability (R)

**Type** ***JxfsPtrCtrlTurnCapability***
**Initial Value** see *JxfsPtrCtrlTurnCapability*
**Description** This property defines the printer's turning media capabilities.

### 4.6.3 Methods

#### atpBackward

**Syntax** ***identificationID atpBackward() throws JxfsException;***
**Description** This command is used to turn the page of the passbook backward.
**Parameter** **none**
**Exceptions** No additional exceptions generated.
**Events** Following events can be generated :

**OperationCompleteEvent**
When a *atpBackward()* operation is completed an *OperationCompleteEvent* will be sent by J/XFS Printer Device Control to all registered listeners with the following data:

| Field | Value |
|---|---|
| *operationID* | JXFS_O_PTR_ATP_BACKWARD |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_MEDIA_JAM<br>JXFS_E_PTR_NO_MEDIA_PRESENT |
| *data* | none |

**StatusEvent**
When the status of the media changes a *StatusEvent* is sent to all registered listeners with the following data :

| Field | Value |
|---|---|
| *status* | JXFS_S_PTR_MEDIA |
| *details* | JxfsMediaStatus mediaStatus<br>The new media status. |

#### atpForward

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

**Description** This command is used to turn the page of the passbook forward.
**Parameter** **none**
**Exceptions** No additional exceptions generated.
**Events** Following events can be generated :

**OperationCompleteEvent**
When a *atpForward()* operation is completed an *OperationCompleteEvent* will be sent by J/XFS Printer Device Control to all registered listeners with the following data:

| **Field** | **Value** |
|---|---|
| *operationID* | JXFS_O_PTR_ATP_FORWARD |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_MEDIA_JAM<br>JXFS_E_PTR_NO_MEDIA_PRESENT |
| *data* | none |

**StatusEvent**
When the status of the media changes a *StatusEvent* is sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_MEDIA |
| *details* | JxfsMediaStatus mediaStatus<br>The new media status. |

### turnMedia

**Syntax** ***identificationID turnMedia() throws JxfsException;***
**Description** This command is used to turn the inserted media.
**Parameter** **none**
**Exceptions** No additional exceptions generated.
**Events** Following events can be generated :

**OperationCompleteEvent**
When a *turnMedia()* operation is completed an *OperationCompleteEvent* will be sent by J/XFS Printer Device Control to all registered listeners with the following data:

| **Field** | **Value** |
|---|---|
| *operationID* | JXFS_O_PTR_ TURN_MEDIA |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_MEDIA_JAM<br>JXFS_E_PTR_MEDIA_TURN_FAIL<br>JXFS_E_PTR_NO_MEDIA_PRESENT |
| *data* | none |

**StatusEvent**
When the status of the media changes a *StatusEvent* is sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_MEDIA |
| *details* | JxfsMediaStatus mediaStatus<br>The new media status. |

## 4.7 IJxfsRead

### 4.7.1 Summary

| Property | Type | Access |
|---|---|---|
| lampStatus | **JxfsPtrLampStatus** | R |
| readStatusCapability | **JxfsPtrReadStatusCapability** | R |
| readFormCapability | **JxfsPtrReadFormCapability** | R |
| readImageCapability | **JxfsPtrReadImageCapability** | R |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| readForm | identificationID |
| readForm *(deprecated)* | identificationID |
| readImage | identificationID |
| readImage *(deprecated)* | identificationID |

| Event | May occur during / after |
|---|---|
| StatusEvent | |
| JXFS_S_PTR_LAMP | *readForm(), readImage()* |
| JXFS_S_PTR_MEDIA | *readForm(), readImage()* |
| IntermediateEvent | |
| JXFS_I_PTR_NO_MEDIA_PRESENT | *readForm(), readImage()* |
| OperationCompleteEvent | |
| JXFS_O_PTR_READ_FORM_DATA | *readForm()* |
| JXFS_O_PTR_READ_IMAGE | *readImage()* |

### 4.7.2 Properties

#### lampStatus (R)

**Type** ***JxfsPtrLampStatus***
**Initial Value** see *JxfsPtrLampStatus*
**Description** This property represents the scanner's imaging lamp status.

#### readStatusCapability (R)

**Type** ***JxfsPtrReadStatusCapability***
**Initial Value** see *JxfsPtrReadStatusCapability*
**Description** This property defines the printer's capability to determine the status of the reading components.

#### readFormCapability (R)

**Type** ***JxfsPtrReadFormCapability***
**Initial Value** see *JxfsPtrReadFormCapability*
**Description** This property defines the printer's form reading capabilities.

#### readImageCapability (R)

**Type** ***JxfsPtrReadImageCapability***
**Initial Value** see *JxfsPtrReadImageCapability*
**Description** This property defines the printer's image reading capabilities.

### 4.7.3 Methods

#### readForm

**Syntax** ***identificationID readForm(String formName, String mediaName, String[] fieldNames) throws JxfsException***
**Description** This method reads fields specified in the *fieldNames* array from the form with the name *formName* using the media description defined by *mediaName.* After a successful completion of this input operation, a

*OperationCompleteEvent* is issued to inform the application of the results. If no media is present the printer should wait enlessly for media to be inserted, or until cancelled by the application.

| **Parameter** | **Type** | **Name** | **Meaning** |
|---|---|---|---|
| | String | formName | Name of the form to be read. |
| | String | mediaName | Name of the media containing the form which should be read. If the printer detects a media of a different type, a JXFS_E_PTR_FORM_INVALID error is reported via *OperationCompleteEvent*. |
| | String[] | fieldNames | An array of strings representing names of the fields which should be read. An empty array means that all readable fields in the form should be read. If the field is an index field, then the syntax of the field name is "<FieldName>[<index>]" where <index> indicates the zero based element of the index field. This array is not allowed to contain *null* entries. |

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated:

**OperationCompleteEvent**

When a *readForm()* operation is completed an *OperationCompleteEvent* will be sent by J/XFS Printer Device Control to all registered listeners with the following data:

| | |
|---|---|
| *operationID* | JXFS_O_PTR_READ_FORM_DATA |
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_FIELD_ERROR<br>JXFS_E_PTR_FIELD_NOT_FOUND<br>JXFS_E_PTR_FIELD_SPEC_FAILURE<br>JXFS_E_PTR_FORM_INVALID<br>JXFS_E_PTR_FORM_NOT_FOUND<br>JXFS_E_PTR_MEDIA_INVALID<br>JXFS_E_PTR_MEDIA_JAM<br>JXFS_E_PTR_MEDIA_NOT_FOUND<br>JXFS_E_PTR_MEDIA_OVERFLOW<br>JXFS_E_PTR_MEDIA_SKEWED |
| *data* | String [] readData<br>Set to an array of "<FieldName>=<FieldValue>" strings. If the field is an index field, then the syntax of the entry is "<FieldName>[<index>]=<FieldValue>" where <index> indicates the zero based element of the index field. |

**IntermediateEvent**

If no media is present the J/XFS Printer Device Control will send an *IntermediateEvent* to all registered listeners with the following data :

| | |
|---|---|
| *operationID* | JXFS_O_PTR_READ_FORM_DATA |
| *identificationID* | The corresponding ID |
| *reason* | JXFS_I_PTR_NO_MEDIA_PRESENT |
| *data* | none |

**IntermediateEvent**

If media is inserted and the operation can continue the J/XFS Printer Device Control will send an *IntermediateEvent* to all registered listeners with the following data :

| | |
|---|---|
| *operationID* | JXFS_O_PTR_READ_FORM_DATA |

| | |
|---|---|
| *identificationID* | The corresponding ID |
| *reason* | JXFS_I_PTR_MEDIA_INSERTED |
| *data* | none |

**IntermediateEvent**
If a field error occurs during reading the field data and the Device Service is capable to continue with reading the further fields[2], an *IntermediateEvent* will be sent to all registered listeners with the following data:

| | |
|---|---|
| *operationID* | JXFS_O_PTR_READ_FORM_DATA |
| *identificationID* | The corresponding ID |
| *reason* | JXFS_I_PTR_FIELD_FAILURE |
| *data* | JxfsPtrFieldFailure failure<br>More detailed information about the failure. |

**StatusEvent**
When the status of the media changes a *StatusEvent* is sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_MEDIA |
| *details* | JxfsMediaStatus mediaStatus<br>The new media status. |

**StatusEvent**
When the status of the scanner's imaging lamp changes a *StatusEvent* is sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_LAMP |
| *details* | JxfsPtrLampStatus lampStatus<br>The new lamp status. |

### readForm

**Syntax** ***identificationID readForm(formName) throws JxfsException***

**Description** This method is deprecated. The *readForm()* method with 3 parameters should be used instead.
Please consult CWA 13937-6:2000 E for the specification of this method.

### readImage

**Syntax** ***identificationID readImage(String formName, String mediaName, String[] fieldNames) throws JxfsException;***

**Description** This method is used to read image data from. the form with the name *formName* using the description of the media defined by *mediaName.* After a successful completion of this input operation, an *OperationCompleteEvent* is issued to inform the application of the results. If no media is present the printer should wait endlessly for media to be inserted or until cancelled by the application.

**Parameter**

| **Type** | **Name** | **Meaning** |
|---|---|---|
| String | formName | Name of the form to be read. |
| String | mediaName | Name of the media containing the form which should be read. If the printer detects a media of a different type, a JXFS_E_PTR_FORM_INVALID error is reported via *OperationCompleteEvent*. |
| String[] | fieldNames | An array of strings representing names of the fields which should be read as images. An empty array means that all fields should be read. If the field is an index field, then the syntax of the field |

[2] An abrupt termination of the form reading may be defined by some device specific conditions.

name is "<FieldName>[<index>]", where <index> indicates the zero based element of the index field. This array is not allowed to contain *null* entries.

**Exceptions** No additional exceptions generated.

**Events** Following events can be generated:

**OperationCompleteEvent**
When a *readImage()* operation is completed an *OperationCompleteEvent* will be sent by J/XFS Printer Device Control to all registered listeners with the following data:

| | |
|---|---|
| *identificationID* | The corresponding ID |
| *result* | JXFS_RC_SUCCESSFUL<br>JXFS_E_PTR_FIELD_ERROR<br>JXFS_E_PTR_FIELD_NOT_FOUND<br>JXFS_E_PTR_FIELD_SPEC_FAILURE<br>JXFS_E_PTR_FORM_INVALID<br>JXFS_E_PTR_FORM_NOT_FOUND<br>JXFS_E_PTR_MEDIA_INVALID<br>JXFS_E_PTR_MEDIA_JAM<br>JXFS_E_PTR_MEDIA_NOT_FOUND<br>JXFS_E_PTR_MEDIA_OVERFLOW<br>JXFS_E_PTR_MEDIA_SKEWED |
| *data* | JxfsPtrImage[] readData<br>An array of images successfully read by this operation. |

**IntermediateEvent**
If no media is present the J/XFS Printer Device Control will send an *IntermediateEvent* to all registered listeners with the following data :

| | |
|---|---|
| *operationID* | JXFS_O_PTR_READ_IMAGE |
| *identificationID* | The corresponding ID |
| *reason* | JXFS_I_PTR_NO_MEDIA_PRESENT |
| *data* | none |

**IntermediateEvent**
If media is inserted and the operation can continue the J/XFS Printer Device Control will send an *IntermediateEvent* to all registered listeners with the following data :

| | |
|---|---|
| *operationID* | JXFS_O_PTR_READ_IMAGE |
| *identificationID* | The corresponding ID |
| *reason* | JXFS_I_PTR_MEDIA_INSERTED |
| *data* | none |

**IntermediateEvent**
If a field error occurs during reading the image data and the Device Service is capable ro continue with reading the further fields[3], an *IntermediateEvent* will be sent to all registered listeners with the following data:

| | |
|---|---|
| *operationID* | JXFS_O_PTR_READ_FORM_DATA |
| *identificationID* | The corresponding ID |
| *reason* | JXFS_I_PTR_FIELD_FAILURE |
| *data* | JxfsPtrFieldFailure failure<br>More detailed information about the failure. |

**StatusEvent**
When the status of the media changes a *StatusEvent* is sent to all registered listeners with the following data:

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_MEDIA |

[3] An abrupt termination of the image reading may be defined by some device specific conditions.

| | |
|---|---|
| *details* | JxfsMediaStatus mediaStatus<br>The new media status. |

**StatusEvent**
When the status of the scanner's imaging lamp changes a *StatusEvent* is sent to all registered listeners with the following data :

| **Field** | **Value** |
|---|---|
| *status* | JXFS_S_PTR_LAMP |
| *details* | JxfsPtrLampStatus lampStatus<br>The new lamp status. |

### readImage

**Syntax** ***identificationID readImage() throws JxfsException***

**Description** This method is deprecated. The *readImage()* method with 3 parameters should be used instead.
Please consult CWA 13937-6:2000 E for the specification of this method.

# 5 Support Classes

### Summary

| Class | Description |
|---|---|
| JxfsPtrCtrlMediaCapability | Specifies Control Media capabilities. |
| JxfsPtrCtrlTurnCapability | Specifies if the printer is able to turn media. |
| JxfsPtrEjectStatusCapability | Specifies the printer's capabilities to determine states of it's ejecting components. |
| JxfsPtrExtentCapability | Specifies if the printer is able to measure the extent of the media. |
| JxfsPtrField | Specifies the description of a field. |
| JxfsPtrFieldFailure | Specifies the failure that occurred during field processing during *writeForm()*, *readForm()* or *readImage()* operations. |
| JxfsPtrForm | Specifies the description of a form. |
| JxfsPtrFormsConfig | Specifies the configuration necessary to print a form. |
| JxfsPtrImage | Specifies the data of the read image. |
| JxfsPtrMaxRetractCapability | Specifies maximum retract capabilities. |
| JxfsPtrMaxStackerCapability | Specifies maximum stacker capabilities. |
| JxfsPtrMedia | Specifies the description of a media. |
| JxfsPtrReadStatusCapability | Specifies the printer's capabilities to determine states of it's reading components. |
| JxfsPtrReadFormCapability | Specifies read form capabilities. |
| JxfsPtrReadImageCapability | Specifies read image capabilities. |
| JxfsPtrStackerCount | Specifies the stacker counter. |
| JxfsPtrStatusCapability | Specifies the printer capabilities to read states of it's components. |
| JxfsPtrRetractCount | Specifies the retract counter. |
| JxfsPtrWriteFormCapability | Specifies write form capabilities. |

## 5.2 JxfsPtrCtrlMediaCapability

This class specifies the control media capabilities of the printer.

### 5.2.1 Summary

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

| Property | Type | Access |
|---|---|---|
| ctrlMediaCapability | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrCtrlMediaCapability | ctrlMediaCapability | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| isCtrlAlarmSupported | boolean |
| isCtrlStampSupported | boolean |
| isCtrlCutSupported | boolean |
| isCtrlEjectSupported | boolean |
| isCtrlFlushSupported | boolean |
| isCtrlPartialCutSupported | boolean |
| isCtrlPerforateSupported | boolean |
| isCtrlRetractSupported | boolean |
| isCtrlSkipSupported | boolean |
| isCtrlStackSupported | boolean |

### 5.2.2 Properties

#### ctrlMediaCapability (R)

**Type** ***int***
**Initial Value** 0
**Description** Specifies the manner in which media can be controlled, as a combination of the following bit flags:
JXFS_PTR_CTRL_ALARM
JXFS_PTR_CTRL_STAMP
JXFS_PTR_CTRL_CUT
JXFS_PTR_CTRL_EJECT
JXFS_PTR_CTRL_FLUSH
JXFS_PTR_CTRL_PARTIALCUT
JXFS_PTR_CTRL_PERFORATE
JXFS_PTR_CTRL_RETRACT
JXFS_PTR_CTRL_SKIP
JXFS_PTR_CTRL_STACK

### 5.2.3 Methods

#### isCtrlAlarmSupported

**Syntax** ***boolean isCtrlAlarmSupported() throws JxfsException;***
**Description** Returns TRUE if the printer has the capability to issue an alarm (the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_ALARM).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

#### isCtrlStampSupported

**Syntax** ***boolean isCtrlStampSupported() throws JxfsException;***
**Description** Returns TRUE if the printer has the capability to stamp on the media

(the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_STAMP).

| | |
|---|---|
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isCtrlCutSupported

| | |
|---|---|
| **Syntax** | ***boolean isCtrlCutSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has the capability to cut the media (the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_CUT). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isCtrlEjectSupported

| | |
|---|---|
| **Syntax** | ***boolean isCtrlEjectSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has the capability to eject the media (the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_EJECT). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isCtrlFlushSupported

| | |
|---|---|
| **Syntax** | ***boolean isCtrlFlushSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has the capability to store data internally and then print it after a flush (the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_FLUSH). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isCtrlPartialCutSupported

| | |
|---|---|
| **Syntax** | ***boolean isCtrlPartialCutSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has the capability to cut the media partially (the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_PARTIALCUT). A partially cut paper is very lose connected to the rest of the media and can very easily be ripped off by the customer. |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isCtrlPerforateSupported

| | |
|---|---|
| **Syntax** | ***boolean isCtrlPerforateSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has the capability to perforate the media (the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_PERFORATE). Perforated media is harder to rip off by the customer than the one which was partially cut. |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isCtrlRetractSupported

| | |
|---|---|
| **Syntax** | ***boolean isCtrlRetractSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has the capability to retract the media (the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_RETRACT). |
| **Parameter** | **None** |

| | |
|---|---|
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isCtrlSkipSupported

| | |
|---|---|
| **Syntax** | ***boolean isCtrlSkipSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has the capability to skip the media to the next mark (the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_SKIP). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isCtrlStackSupported

| | |
|---|---|
| **Syntax** | ***boolean isCtrlStackSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has the capability to stack the media (the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_STACK). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

## 5.3 JxfsPtrCtrlTurnCapability

This class specifies the turn media capabilities of the printer.

### 5.3.1 Summary

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

| Property | Type | Access |
|---|---|---|
| ctrlTurnMediaCapability | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrCtrlTurnCapability | ctrlTurnMediaCapability | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| isCtrlATPBackwardSupported | boolean |
| isCtrlATPForwardSupported | boolean |
| isCtrlMediaTurnSupported | boolean |

### 5.3.2 Properties

#### ctrlTurnMediaCapability (R)

**Type** ***int***
**Initial Value** 0
**Description** Specifies the manner in which media can be controlled, as a combination of the following bit flags:
JXFS_PTR_CTRL_ATP_BACKWARD
JXFS_PTR_CTRL_ATP_FORWARD
JXFS_PTR_CTRL_TURNMEDIA

### 5.3.3 Methods

#### isCtrlATPBackwardSupported

**Syntax** ***boolean isCtrlBackwardSupported() throws JxfsException;***
**Description** Returns TRUE if the printer has the capability to turn one page backward (the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_ATP_BACKWARD).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

#### isCtrlATPForwardSupported

**Syntax** ***boolean isCtrlATPForwardSupported() throws JxfsException;***
**Description** Returns TRUE if the printer has the capability to turn one page forward(the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_ATP_FORWARD).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

#### isCtrlTurnMediaSupported

**Syntax** ***boolean isCtrlTurnMediaSupported() throws JxfsException;***
**Description** Returns TRUE if the printer has the capability to turn the media (the *ctrlMediaCapability* property contains the value JXFS_PTR_CTRL_TURNMEDIA).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

## 5.4 JxfsPtrEjectStatusCapability

This class specifies the printer's capabilities to determine states of it's ejecting components.

### 5.4.1 Summary

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

| Property | Type | Access |
|---|---|---|
| ejectStatusCapability | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrEjectStatusCapability | ejectStatusCapability | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| isExitEntrySupported | boolean |
| isInkSupported | boolean |
| isStackerSupported | boolean |

### 5.4.2 Properties

#### ejectStatusCapability (R)

**Type** ***int***
**Initial Value** 0
**Description** Specifies the printer 's capabilities to determine states of it's ejecting components, as a combination of the following bit flags:
JXFS_PTR_STATUS_EXIT_ENTRY
JXFS_PTR_STATUS_INK
JXFS_PTR_STATUS_STACKER

### 5.4.3 Methods

#### isExitEntrySupported

**Syntax** ***boolean isExitEntrySupported() throws JxfsException***
**Description** Returns TRUE if the printer has the capability to determine the status of the exit / entry slot (the *ejectStatusCapability* property contains the value JXFS_PTR_STATUS_EXIT_ENTRY).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

#### isInkSupported

**Syntax** ***boolean isInkSupported() throws JxfsException***
**Description** Returns TRUE if the printer has the capability to determine the status of the stamping ink cartridge (the *ejectStatusCapability* property contains the value JXFS_PTR_STATUS_INK).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

#### isStackerSupported

**Syntax** ***boolean isStackerSupported() throws JxfsException***
**Description** Returns TRUE if the printer has the capability to determine the status of the stacker (the *ejectStatusCapability* property contains the value JXFS_PTR_STATUS_STACKER).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.

| | |
|---|---|
| **Event** | No additional events are generated. |

## 5.5 JxfsPtrExtentCapability

This class specifies the extent capability of the printer.

### 5.5.1 Summary

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

| Property | Type | Access |
|---|---|---|
| extentCapability | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrExtentCapability | extentCapability | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| isExtHorizontalSupported | boolean |
| isExtVerticalSupported | boolean |

### 5.5.2 Properties

#### extentCapability (R)

**Type** ***int***
**Initial Value** 0
**Description** Specifies whether the device is able to measure the inserted media. Depending on the device capability *extentCapability* will be set as a combination of the following values:

| Value | Meaning |
|---|---|
| JXFS_PTR_EXT_HORIZONTAL | Device has horizontal size detection capability. |
| JXFS_PTR_EXT_VERTICAL | Device has vertical size detection capability. |

### 5.5.3 Methods

#### isExtHorizontalSupported

**Syntax** ***boolean isExtHorizontalSupported() throws JxfsException;***
**Description** Returns TRUE if the printer is able to measure the horizontal size of the inserted media (the *extentCapability* property contains the value JXFS_PTR_EXT_HORIZONTAL)
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

#### isExtVerticalSupported

**Syntax** ***boolean isExtVerticalSupported() throws JxfsException;***
**Description** Returns TRUE if the printer is able to measure the vertical size of the inserted media (the *extentCapability* property contains the value JXFS_PTR_EXT_VERTICAL)
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

## 5.6 JxfsPtrField

The JxfsPtrField class contains the properties of a field on a specified form.

### 5.6.1 Summary

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

| Property | Type | Access |
|---|---|---|
| access | int | R |
| classType | int | R |
| fieldName | String | R |
| format | String | R |
| indexCount | int | R |
| initialValue | String | R |
| overflow | int | R |
| type | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrField | access | int |
| | classType | int |
| | fieldName | String |
| | format | String |
| | indexCount | int |
| | initialValue | String |
| | overflow | int |
| | type | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |

### 5.6.2 Properties

**access (R)**

**Type** ***int***
**Initial Value** 0
**Description** Indicates whether the field is to be used for input, output or both and can be a combination of the following values:

| Value | Meaning |
|---|---|
| JXFS_PTR_FRM_ACCESS_READ | Field is used for input. |
| JXFS_PTR_FRM_ACCESS_WRITE | Field is used for output. |

**classType (R)**

**Type** ***int***
**Initial Value** JXFS_PTR_FRM_CLASS_OPTIONAL
**Description** Indicates the class of the field and can be one of the following values:

| Value | Meaning |
|---|---|
| JXFS_PTR_FRM_CLASS_STATIC | Field data cannot be set by the application. |
| JXFS_PTR_FRM_CLASS_OPTIONAL | Field data can be set by the application. |
| JXFS_PTR_FRM_CLASS_REQUIRED | Field data must be set by the application |

**fieldName (R)**

**Type** ***String***

**Initial Value** empty String
**Description** Name of the field, unique in the scope of a form.

### format (R)

**Type** ***String***
**Initial Value** empty String
**Description** Indicates the format as defined in the form for this field. The application can use this field for application-specific formatting of the field value. For example, a "%f10.3" could be a C-style formatting string for printing a float.
The value of this property doesn't affect the way in which the field is printed. The usage of this property by the application is strongly discouraged, because it may lead to many incompatibilities between different Device Service implementations.

### indexCount (R)

**Type** ***int***
**Initial Value** 0
**Description** Indicates the number of entries for an index field. A value of zero indicates that this field is not an index field. Index fields are typically used to present information in a tabular fashion.

### initialValue (R)

**Type** ***String***
**Initial Value** empty String
**Description** Indicates the initial value of the field. When the form is printed, this value will be used if another value is not provided.

### overflow (R)

**Type** ***int***
**Initial Value** JXFS_PTR_FRM_OVF_TRUNCATE
**Description** Indicates how an overflow of the field data should be handled and can be one of the following values:

| Value | Meaning |
|---|---|
| JXFS_PTR_FRM_OVF_TERMINATE | Return an error and terminate printing the form. |
| JXFS_PTR_FRM_OVF_TRUNCATE | Truncate field data to fit in the field. |
| JXFS_PTR_FRM_OVF_BEST_FIT | Fit text in the field. |
| JXFS_PTR_FRM_OVF_OVERWRITE | Print field data beyond the extents of the field boundary. |
| JXFS_PTR_FRM_OVF_WORDWRAP | If field can hold more than one line the text is wrapped around. |

### type (R)

**Type** ***int***
**Initial Value** 0
**Description** Indicates the type of the field and can be one of the following values:

| Value | Meaning |
|---|---|
| JXFS_PTR_FRM_FIELD_BARCODE | Barcode field. |
| JXFS_PTR_FRM_FIELD_GRAPHIC | Graphic field. |
| JXFS_PTR_FRM_FIELD_MICR | Magnetic Ink Character Recognition field. |
| JXFS_PTR_FRM_FIELD_MSF | Magnetic Stripe Facility field. |
| JXFS_PTR_FRM_FIELD_OCR | Optical Recognition Character field. |

| | |
|---|---|
| JXFS_PTR_FRM_FIELD_PAGEMARK | Page Mark field. |
| JXFS_PTR_FRM_FIELD_TEXT | Text field. |

## 5.7 JxfsPtrFieldFailure

Instances of this class are returned as detailed description of JXFS_I_FIELD_FAILURE intermediate events.

### 5.7.1 Summary

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

| Property | Type | Access |
|---|---|---|
| fieldFailure | int | R |
| fieldName | String | R |
| formName | String | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrFieldFailure | fieldFailure | int |
| | fieldName | String |
| | formName | String |

| Method | Return | May be used after |
|---|---|---|
| get*Property* | *Property* | |

### 5.7.2 Properties

#### fieldFailure (R)

**Type** ***int***
**Initial Value** 0
**Description** Specifies the type of failure and can be one of the following:
JXFS_E_PTR_FIELD_GRAPHIC
JXFS_E_PTR_FIELD_HW_ERROR
JXFS_E_PTR_FIELD_NOT_READ
JXFS_E_PTR_FIELD_NOT_WRITE
JXFS_E_PTR_FIELD_OVERFLOW
JXFS_E_PTR_FIELD_REQUIRED
JXFS_E_PTR_FIELD_STATIC_OVWR
JXFS_E_PTR_FIELD_TYPE_NOT_SUPPORTED

#### fieldName (R)

**Type** ***String***
**Initial Value** empty String
**Description** Specifies the name of the field at which the error occurred. If the field is an indexed field its name will be in the format "<fieldName>[<index>]".

#### formName (R)

**Type** ***String***
**Initial Value** empty String
**Description** Specifies the name of the form at which the error occurred

## 5.8 JxfsPtrForm

The JxfsPtrForm class contains the properties of a specified form.

### 5.8.1 Summary

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

| Property | Type | Access |
|---|---|---|
| alignment | int | R |
| base | int | R |
| fields | String[] | R |
| formName | String | R |
| height | int | R |
| offsetX | int | R |
| offsetY | int | R |
| orientation | int | R |
| unitX | int | R |
| unitY | int | R |
| userPrompt | String | R |
| versionMajor | int | R |
| versionMinor | int | R |
| width | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrForm | alignment | int |
| | base | int |
| | fields | String[] |
| | formName | String |
| | height | int |
| | offsetX | int |
| | offsetY | int |
| | orientation | int |
| | unitX | int |
| | unitY | int |
| | userPrompt | String |
| | versionMajor | int |
| | versionMinor | int |
| | width | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |

## 5.8.2 Properties

**alignment (R)**

**Type** ***int***
**Initial Value** 0
**Description** Indicates the relative alignment of the form on the media as one of the following values :

| Value | Meaning |
|---|---|
| JXFS_PTR_ALN_TOPLEFT | Align the form to top left of media |
| JXFS_PTR_ALN_TOPRIGHT | Align the form to top right of media |
| JXFS_PTR_ALN_BOTTOMLEFT | Align the form to bottom left of media |
| JXFS_PTR_ALN_BOTTOMRIGHT | Align the form to bottom right of media |

**base (R)**

**Type** ***int***
**Initial Value** JXFS_PTR_FRM_MM
**Description** Indicates the base unit of measurement of the form as one of the following values:

| Value | Meaning |
|---|---|
| JXFS_PTR_FRM_INCH | Base unit is inches. |
| JXFS_PTR_FRM_MM | Base unit is millimeters. |
| JXFS_PTR_FRM_ROWCOLUMN | Base unit is rows and columns. |

**fields (R)**

**Type** ***String[]***
**Initial Value** empty String[]
**Description** Indicates the field names on the form

**formName (R)**

**Type** ***String***
**Initial Value** empty String
**Description** Indicates the name of the form.

**height (R)**

**Type** ***int***
**Initial Value** 0
**Description** Indicates the height of the form in terms of the base vertical resolution.

**offsetX (R)**

**Type** ***int***
**Initial Value** 0
**Description** For JXFS_PTR_ALN_TOPLEFT and JXFS_PTR_ALN_BOTTOMLEFT *alignment* values: this value indicates the horizontal offset of the form's left edge position, relative to the left edge of the media.

For JXFS_PTR_ALN_TOPRIGHT and JXFS_PTR_ALN_BOTTOMRIGHT *alignment* values: this value indicates the horizontal offset of the form's right edge position, relative to the right edge of the media.

This value is specified in terms of the unitX property and is always positive.

**offsetY (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | For JXFS_PTR_ALN_TOPLEFT and JXFS_PTR_ALN_TOPRIGHT *alignment* values: this value indicates the vertical offset of the form's top edge position, relative to the top edge of the media.<br><br>For JXFS_PTR_ALN_BOTTOMLEFT and JXFS_PTR_ALN_BOTTOMRIGHT *alignment* values: this value indicates the vertical offset of the form's bottom edge position, relative to the bottom edge of the media.<br><br>This value is specified in terms of the unitY property and is always positive. |

**orientation (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | JXFS_PTR_FRM_PORTRAIT |
| **Description** | Indicates the orientation of the form and can be one of the following values : |

| **Value** | **Meaning** |
|---|---|
| JXFS_PTR_FRM_LANDSCAPE | Orientation of the form is landscape. |
| JXFS_PTR_FRM_PORTRAIT | Orientation of the form is portrait. |

**unitX (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 1 |
| **Description** | Indicates the horizontal resolution of the base units as a fraction of the *base* value. This property should be interpreted as a denominator with a numerator of 1. So, for example, if the *base* property contains the value JXFS_PTR_FRM_MM and the *unitX* the value 10, a value 20 for the property *offsetX* should be interpreted as 2 mm. |

**unitY (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 1 |
| **Description** | Indicates the vertical resolution of the base units as a fraction of the *base* value. This property should be interpreted as a denominator with a numerator of 1. So, for example, if the *base* property contains the value JXFS_PTR_FRM_MM and the *unitY* the value 10, a value 20 for the property *offsetY* should be interpreted as 2 mm. |

**userPrompt (R)**

| | |
|---|---|
| **Type** | ***String*** |
| **Initial Value** | empty String |
| **Description** | Indicates the user prompt string. |

**versionMajor (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the major version of the form. |

**versionMinor (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the minor version of the form. |

**width (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the width of the form in terms of the base horizontal resolution. |

## *5.9* JxfsPtrFormsConfig

This class contains properties and methods to configure the usage of forms.

### 5.9.1 Summary

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

| Property | Type | Access |
|---|---|---|
| alignment | int | R/W |
| base | int | R/W |
| formsDescriptionList | **JxfsPtrForm[]** | R/W |
| mediaDescriptionList | **JxfsPtrMedia[]** | R/W |
| offsetX | int | R/W |
| offsetY | int | R/W |
| unitX | int | R/W |
| unitY | int | R/W |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrFormsConfig | *none* | |
| JxfsPtrFormsConfig | alignment | int |
| | base | int |
| | offsetX | int |
| | offsetY | int |
| | unitX | int |
| | unitY | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| set*Property* | void |

### 5.9.2 Properties

#### alignment (R/W)

**Type** ***int***
**Initial Value** JXFS_PTR_ALN_USEFORMDEFN
**Description** Indicates the relative alignment of the form on the media, as one of the following values:

| Value | Meaning |
|---|---|
| JXFS_PTR_ALN_BOTTOMLEFT | Align the form to bottom left of media. |
| JXFS_PTR_ALN_BOTTOMRIGHT | Align the form to bottom right of media. |
| JXFS_PTR_ALN_TOPLEFT | Align the form to top left of media. |
| JXFS_PTR_ALN_TOPRIGHT | Align the form to top right of media. |
| JXFS_PTR_ALN_USEFORMDEFN | Use alignment specified in the form definition. |

#### base (R/W)

**Type** ***int***
**Initial Value** JXFS_PTR_FRM_MM
**Description** Indicates the base unit of measurement of the media and can be one of the following values:

| Value | Meaning |
|---|---|
| JXFS_PTR_FRM_INCH | Base unit is inches. |
| JXFS_PTR_FRM_MM | Base unit is millimeters. |
| JXFS_PTR_FRM_ROWCOLUMN | Base unit is rows and columns. |

### formsDescriptionList (R/W)

This property is deprecated. It is mentioned here for compatibility reasons only. Getting this property returns an empty array. Setting this property has no effect.

### mediaDescriptionList (R/W)

This property is deprecated. It is mentioned here for compatibility reasons only. Getting this property returns an empty array. Setting this property has no effect.

### offsetX (R/W)

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | JXFS_PTR_OFFSET_USEFORMDEFN |
| **Description** | For JXFS_PTR_ALN_TOPLEFT and JXFS_PTR_ALN_BOTTOMLEFT *alignment* values: this value indicates the horizontal offset of the form's left edge position, relative to the left edge of the media.<br><br>For JXFS_PTR_ALN_TOPRIGHT and JXFS_PTR_ALN_BOTTOMRIGHT *alignment* values: this value indicates the horizontal offset of the form's right edge position, relative to the right edge of the media.<br><br>This value is specified in terms of the unitX property and is always positive. A value of JXFS_PTR_OFFSET_USEFORMDEFN specifies that the *offsetX* from the form definition should be used. |

### offsetY (R/W)

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | JXFS_PTR_OFFSET_USEFORMDEFN |
| **Description** | For JXFS_PTR_ALN_TOPLEFT and JXFS_PTR_ALN_TOPRIGHT *alignment* values: this value indicates the vertical offset of the form's top edge position, relative to the top edge of the media.<br><br>For JXFS_PTR_ALN_BOTTOMLEFT and JXFS_PTR_ALN_BOTTOMRIGHT *alignment* values: this value indicates the vertical offset of the form's bottom edge position, relative to the bottom edge of the media.<br><br>This value is specified in terms of the unitY property and is always positive. A value of JXFS_PTR_OFFSET_USEFORMDEFN specifies that the *offsetY* from the form definition should be used. |

### unitX (R/W)

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 1 |
| **Description** | Indicates the horizontal resolution of the base units as a fraction of the property *base.* This property should be interpreted as a denominator with a numerator of 1. So, for example, if the *base* property contains the value JXFS_PTR_FRM_MM and the *unitX* the value 10, a value 20 for the property *offsetX* should be interpreted as 2 mm. |

### unitY (R/W)

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 1 |
| **Description** | Indicates the vertical resolution of the base units as a fraction of the property *base.* This property should be interpreted as a denominator with a numerator of 1. So, for example, if the *base* property contains the value JXFS_PTR_FRM_MM and the *unitY* the value 10, a value 20 for the property *offsetY* should be interpreted as 2 mm. |

## 5.10 JxfsPtrImage

This class specifies the data of the image read by the readImage method.

### 5.10.1 Summary

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

| Property | Type | Access |
|---|---|---|
| fieldName | String | R |
| imageData | byte[] | R |
| imageType | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrImage | fieldName | String |
| | imageData | byte[] |
| | imageType | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |

### 5.10.2 Properties

#### fieldName (R)

**Type** ***String***
**Initial Value** empty String
**Description** Indicates the name of the field within the form.

#### imageData (R)

**Type** ***byte[]***
**Initial Value** empty byte[]
**Description** Image data from the current media.

#### imageType (R)

**Type** ***int***
**Initial Value** 0
**Description** Set to the image data format and can be one of the following values:
**JXFS_PTR_IMAGE_TIF**
Image data is in TIF format.
**JXFS_PTR_IMAGE_MTF**
Image data is in MTF format.
**JXFS_PTR_IMAGE_BMP**
Image data is in BMP format.

## 5.11 JxfsPtrMaxRetractCapability

This class specifies the maximum possible number of retracts the printer can perform.

### 5.11.1 Summary

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

| Property | Type | Access |
|---|---|---|
| maxRetractCapability | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrMaxRetractCapability | maxRetractCapability | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |

### 5.11.2 Properties

#### maxRetractCapability (R)

**Type** ***int***
**Initial Value** 0
**Description** Specifies the maximum number of media items that the retract bin can hold (zero if not available).

## 5.12 JxfsPtrMaxStackerCapability

This class defines the maximum number of media items that the stacker can hold.

### 5.12.1 Summary

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

| Property | Type | Access |
|---|---|---|
| maxStackerCapability | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrMaxStackerCapability | maxStackerCapability | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |

### 5.12.2 Properties

#### maxStackerCapability (R)

**Type** ***int***
**Initial Value** 0
**Description** Specifies the maximum number of media items that the stacker can hold (zero if not available).

## 5.13 JxfsPtrMedia

The JxfsPtrMedia class contains the properties of a specified media.

### 5.13.1 Summary

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

| Property | Type | Access |
|---|---|---|
| base | int | R |
| foldtype | int | R |
| lineCount | int | R |
| mediaName | String | R |
| mediaType | int | R |
| pageCount | int | R |
| printAreaHeight | int | R |
| printAreaWidth | int | R |
| printAreaX | int | R |
| printAreaY | int | R |
| restrictedAreaHeight | int | R |
| restrictedAreaWidth | int | R |
| restrictedAreaX | int | R |
| restrictedAreaY | int | R |
| sizeHeight | int | R |
| sizeWidth | int | R |
| stagger | int | R |
| unitX | int | R |
| unitY | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrMedia | base | int |
| | foldtype | int |
| | lineCount | int |
| | mediaName | String |
| | mediaType | int |
| | pageCount | int |
| | printAreaHeight | int |
| | printAreaWidth | int |
| | printAreaX | int |
| | printAreaY | int |
| | restrictedAreaHeight | int |
| | restrictedAreaWidth | int |
| | restrictedAreaX | int |
| | restrictedAreaY | int |
| | sizeHeight | int |
| | sizeWidth | int |
| | stagger | int |
| | unitX | int |
| | unitY | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |

### 5.13.2 Properties

All references to “base resolution” are always in terms of unitX and unitY properties.

**base (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | JXFS_PTR_FRM_MM |
| **Description** | Indicates the base unit of measurement of the form as one of the following values: |

| **Value** | **Meaning** |
|---|---|
| JXFS_PTR_FRM_INCH | Base unit is inches. |
| JXFS_PTR_FRM_MM | Base unit is millimeters. |
| JXFS_PTR_FRM_ROW COLUMN | Base unit is rows and columns. |

**foldType (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the type of fold for a media of type **JXFS_PTR_FRM_MEDIA_PASSBOOK** and can be one of the following values : |

| **Value** | **Meaning** |
|---|---|
| JXFS_PTR_FRM_FOLD_ HORIZONTAL | Passbook has horizontal fold. |
| JXFS_PTR_FRM_FOLD_ NONE | Passbook has no fold. |
| JXFS_PTR_FRM_FOLD_ VERTICAL | Passbook has vertical fold. |

**lineCount (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the number of lines on a page for media of type **JXFS_PTR_FRM_MEDIA_PASSBOOK**. |

**mediaName (R)**

| | |
|---|---|
| **Type** | ***String*** |
| **Initial Value** | empty String |
| **Description** | Indicates the name of the media. |

**mediaType (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the type of media as one of the following values : |

| **Value** | **Meaning** |
|---|---|
| JXFS_PTR_FRM_MEDIA_ GENERIC | Generic media, i.e., single sheet. |
| JXFS_PTR_FRM_MEDIA_ MULTIPART | Multipart media. |
| JXFS_PTR_FRM_MEDIA_ PASSBOOK | Passbook media. |

**pageCount (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the number of pages in a passbook for media of type **JXFS_PTR_FRM_MEDIA_PASSBOOK**. |

**printAreaHeight (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the printable area height of the media in terms of the base |

vertical resolution.

**printAreaWidth (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the printable area width of the media in terms of the base horizontal resolution. |

**printAreaX (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the horizontal offset of the printable area relative to the top left corner of the media in terms of the base horizontal resolution. |

**printAreaY (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the vertical offset of the printable area relative to the top left corner of the media in terms of the base vertical resolution. |

**restrictedAreaHeight (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the restricted area height of the media in terms of the base vertical resolution. |

**restrictedAreaWidth (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the restricted area width of the media in terms of the base horizontal resolution. |

**restrictedAreaX (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the horizontal offset of the restricted area relative to the top left corner of the media in terms of the base horizontal resolution. |

**restrictedAreaY (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the vertical offset of the restricted area relative to the top left corner of the media in terms of the base vertical resolution. |

**sizeHeight (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the height of the media in terms of the base vertical resolution. |

**sizeWidth (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |
| **Description** | Indicates the width of the media in terms of the base horizontal resolution. |

**stagger (R)**

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 0 |

| | |
|---|---|
| **Description** | Indicates the staggering area from the top of the media in terms of the base vertical resolution for a media of type **JXFS_PTR_FRM_MEDIA_PASSBOOK**. |

### unitX (R)

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 1 |
| **Description** | Indicates the horizontal resolution of the base units as a fraction of the property *base.* This property should be interpreted as a denominator with a numerator of 1. So, for example, if the *base* property contains the value JXFS_PTR_FRM_MM and the *unitX* the value 10, a value 20 for the property *offsetX* should be interpreted as 2 mm. |

### unitY (R)

| | |
|---|---|
| **Type** | ***int*** |
| **Initial Value** | 1 |
| **Description** | Indicates the vertical resolution of the base units as a fraction of the property *base.* This property should be interpreted as a denominator with a numerator of 1. So, for example, if the *base* property contains the value JXFS_PTR_FRM_MM and the *unitY* the value 10, a value 20 for the property *offsetY* should be interpreted as 2 mm. |

## 5.14 JxfsPtrMediaExtents

This class contains the properties to return a media's extents.

### 5.14.1 Summary

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

| Property | Type | Access |
|---|---|---|
| sizeX | int | R |
| sizeY | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrMediaExtents | sizeX | int |
| | sizeY | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |

### 5.14.2 Properties

**sizeX (R)**

**Type** ***int***
**Initial Value** 0
**Description** Indicates the width of the media in terms of the base horizontal resolution.

**sizeY (R)**

**Type** ***int***
**Initial Value** 0
**Description** Indicates the height of the media in terms of the base vertical resolution.

## 5.15 JxfsPtrReadFormCapability

This class specifies the read form capabilities of the printer.

### 5.15.1 Summary

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

| Property | Type | Access |
|---|---|---|
| readFormCapability | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrReadFormCapability | readFormCapability | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| isBarcodeReadSupported | boolean |
| isImageReadSupported | boolean |
| isOCRReadSupported | boolean |
| isTextReadSupported | boolean |
| isMICRReadSupported | boolean |
| isMSFReadSupported | boolean |
| isPagemarkReadSupported | boolean |

### 5.15.2 Properties

#### readFormCapability (R)

**Type** ***int***
**Initial Value** JXFS_PTR_READ_TEXT
**Description** Specifies whether the device can read data from the media. Depending on the device capability *readFormCapability* will be set as a combination of the following values:

| Value | Meaning |
|---|---|
| JXFS_PTR_READ_BARCODE | Device has Barcode capability. |
| JXFS_PTR_READ_IMAGE | Device has imaging capability. |
| JXFS_PTR_READ_MICR | Device has MICR capability. |
| JXFS_PTR_READ_MSF | Device has MSF capability. |
| JXFS_PTR_READ_OCR | Device has OCR capability. |
| JXFS_PTR_READ_TEXT | Device has Text capability. |
| JXFS_PTR_READ_PAGEMARK | Device has pagemark capability. |

### 5.15.3 Methods

#### isBarcodeReadSupported

**Syntax** ***boolean isBarcodeReadSupported() throws JxfsException;***
**Description** Returns TRUE if the printer has Barcode capability (the *readFormCapability* property contains the value JXFS_PTR_READ_BARCODE).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

#### isMICRReadSupported

**Syntax** ***boolean isMICRReadSupported() throws JxfsException;***
**Description** Returns TRUE if the printer has MICR capability (the *readFormCapability* property contains the value JXFS_PTR_READ_MICR).
**Parameter** **None**

| | |
|---|---|
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isMSFReadSupported

| | |
|---|---|
| **Syntax** | ***boolean isMSFReadSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has MSF capability (the *readFormCapability* property contains the value JXFS_PTR_READ_MSF). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isOCRReadSupported

| | |
|---|---|
| **Syntax** | ***boolean isOCRReadSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has OCR capability (the *readFormCapability* property contains the value JXFS_PTR_READ_OCR). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isPagemarkReadSupported

| | |
|---|---|
| **Syntax** | ***boolean isPagemarkReadSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has pagemark capability (the *readFormCapability* property contains the value JXFS_PTR_READ_PAGEMARK). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isTextReadSupported

| | |
|---|---|
| **Syntax** | ***boolean isTextReadSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has text reading capability (the *readFormCapability* property contains the value JXFS_PTR_READ_TEXT). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isImageReadSupported

| | |
|---|---|
| **Syntax** | ***boolean isImageReadSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has imaging capability (the *readFormCapability* property contains the value JXFS_PTR_READ_IMAGE). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

## 5.16 JxfsPtrReadImageCapability

This class specifies the read image capabilities of the printer.

### 5.16.1 Summary

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

| Property | Type | Access |
|---|---|---|
| readImageCapability | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrReadImageCapability | readImageCapability | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| isImageTIFSupported | boolean |
| isImageMTFSupported | boolean |
| isImageBMPSupported | boolean |

### 5.16.2 Properties

#### readImageCapability (R)

**Type** ***int***
**Initial Value** 0
**Description** Specifies whether the device can read image data from the media. Depending on the device capability *readImageCapability* will be set as one of the following values:

| Value | Meaning |
|---|---|
| JXFS_PTR_IMAGE_TIF | Device has capability to read tif format. |
| JXFS_PTR_IMAGE_MTF | Device has capability to read mtf format. |
| JXFS_PTR_IMAGE_BMP | Device has capability to read bmp format. |

### 5.16.3 Methods

#### isImageTIFSupported

**Syntax** ***boolean isImageTIFSupported() throws JxfsException;***
**Description** Returns TRUE if the device has the capability to read tif format (the *readImageCapability* property contains the value JXFS_PTR_IMAGE_TIF).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

#### isImageMTFSupported

**Syntax** ***boolean isImageMTFSupported() throws JxfsException;***
**Description** Returns TRUE if the device has the capability to read tif format (the *readImageCapability* property contains the value JXFS_PTR_IMAGE_MTF).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

#### isImageBMPSupported

**Syntax** ***boolean isImageBMPSupported() throws JxfsException;***
**Description** Returns TRUE if the device has the capability to read tif format (the

| | |
|---|---|
| | *readImageCapability* property contains the value JXFS_PTR_IMAGE_BMP). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

## 5.17 JxfsPtrReadStatusCapability

This class specifies the printer's capabilities to determine states of it's reading components.

### 5.17.1 Summary

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

| Property | Type | Access |
|---|---|---|
| readStatusCapability | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrReadStatusCapability | readStatusCapability | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| isLampSupported | boolean |

### 5.17.2 Properties

#### readStatusCapability (R)

**Type** ***int***
**Initial Value** 0
**Description** Specifies the printer 's capabilities to determine states of it's reading components, as a combination of the following bit flags:
JXFS_PTR_STATUS_LAMP

### 5.17.3 Methods

#### isLampSupported

**Syntax** ***boolean isLampSupported() throws JxfsException***
**Description** Returns TRUE if the printer has the capability to determine the status of the scanner's imaging lamp (the *readStatusCapability* property contains the value JXFS_PTR_STATUS_LAMP).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

## 5.18 JxfsPtrStatusCapability

This class specifies the capabilities of the printer to determine states of it's components.

### 5.18.1 Summary

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

| Property | Type | Access |
|---|---|---|
| statusCapability | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrStatusCapability | statusCapability | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| isTonerStatusSupported | boolean |
| isMediaStatusSupported | boolean |
| isPaperStatusSupported | boolean |

### 5.18.2 Properties

#### statusCapability (R)

**Type** ***int***
**Initial Value** 0
**Description** Specifies the capabilities of the printer to determine the states of it's components, as a combination of the following bit flags:
JXFS_PTR_STATUS_TONER
JXFS_PTR_STATUS_MEDIA
JXFS_PTR_STATUS_PAPER

### 5.18.3 Methods

#### isTonerStatusSupported

**Syntax** ***boolean isTonerStatusSupported() throws JxfsException;***
**Description** Returns TRUE if the printer has the capability to determine the toner status.
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

#### isMediaStatusSupported

**Syntax** ***boolean isTonerStatusSupported() throws JxfsException;***
**Description** Returns TRUE if the printer has the capability to determine the media status.
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

#### isPaperStatusSupported

**Syntax** ***boolean isTonerStatusSupported() throws JxfsException;***
**Description** Returns TRUE if the printer has the capability to determine the paper bin status.
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

## 5.19 JxfsPtrRetractCount

This class specifies the number of media the printer has retracted.

## 5.19.1 Summary

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

| Property | Type | Access |
|---|---|---|
| retractCount | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrRetractCount | retractCount | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| set*Property* | void |
| resetRetractCount | void |

## 5.19.2 Properties

### retractCount (R)

**Type** ***int***
**Initial Value** 0
**Description** The number of media retracted; applicable only to printers with retract capability. This value is persistent: It is reset to zero by the *resetRetractCount* method. The *retractCount* can only be set by the Device Service internally.

## 5.19.3 Methods

### resetRetractCount

**Syntax** ***void resetRetractCount() throws JxfsException;***
**Description** Sets the number of retracts to zero.
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

## 5.20 JxfsPtrStackerCount

This class specifies the number of medias the printer has stacked prior to eject.

### 5.20.1 Summary

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

| Property | Type | Access |
|---|---|---|
| stackerCount | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrStackerCount | stackerCount | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| set*Property* | void |
| resetStackerCount | void |

### 5.20.2 Properties

#### stackerCount (R)

**Type** ***int***
**Initial Value** 0
**Description** The number of media stacked; applicable only to printers with stacking capability. This value is persistent: It is reset to zero by the *resetStackerCount* method. The *stackerCount* can only be set by the Device Service internally.

### 5.20.3 Methods

#### resetStackerCount

**Syntax** ***void resetStackerCount() throws JxfsException***
**Description** Sets the number of stacked medias to zero.
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

## 5.21 JxfsPtrWriteFormCapability

This class specifies the write form capabilities of the printer.

### 5.21.1 Summary

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

| Property | Type | Access |
|---|---|---|
| writeFormCapability | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrWriteFormCapability | writeFormCapability | int |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| isBarcodeWriteSupported | boolean |
| isGraphicsWriteSupported | boolean |
| isOCRWriteSupported | boolean |
| isTextWriteSupported | boolean |
| isMICRWriteSupported | boolean |
| isMSFWriteSupported | boolean |

### 5.21.2 Properties

#### writeFormCapability (R)

**Type** ***int***
**Initial Value** JXFS_PTR_WRITE_TEXT
**Description** Specifies whether the device can write data to the media. Depending on the device capability *writeFormCapability* will be set as a combination of the following values:

| Value | Meaning |
|---|---|
| JXFS_PTR_WRITE_BARCODE | Device has Barcode capability. |
| JXFS_PTR_WRITE_GRAPHICS | Device has Graphics capability. |
| JXFS_PTR_WRITE_MICR | Device has MICR capability. |
| JXFS_PTR_WRITE_MSF | Device has MSF capability. |
| JXFS_PTR_WRITE_OCR | Device has OCR capability. |
| JXFS_PTR_WRITE_TEXT | Device has Text capability. |

### 5.21.3 Methods

#### isBarcodeWriteSupported

**Syntax** ***boolean isBarcodeWriteSupported() throws JxfsException;***
**Description** Returns TRUE if the printer has the capability to write barcode to the media (the *writeFormCapability* property contains the value JXFS_PTR_WRITE_BARCODE).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

#### isGraphicsWriteSupported

**Syntax** ***boolean isGraphicsWriteSupported() throws JxfsException;***
**Description** Returns TRUE if the printer has the capability to write graphics to the media (the *writeFormCapability* property contains the value JXFS_PTR_WRITE_GRAPHICS).
**Parameter** **None**
**Exceptions** No additional exceptions are generated.
**Event** No additional events are generated.

### isOCRWriteSupported

| | |
|---|---|
| **Syntax** | ***boolean isOCRWriteSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has the capability to write OCR codes to the media (the *writeFormCapability* property contains the value JXFS_PTR_WRITE_OCR). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isTextWriteSupported

| | |
|---|---|
| **Syntax** | ***boolean isTextWriteSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has the capability to write text to the media (the *writeFormCapability* property contains the value JXFS_PTR_WRITE_TEXT). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isMICRWriteSupported

| | |
|---|---|
| **Syntax** | ***boolean isMICRWriteSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has the capability to write MICR to the media (the *writeFormCapability* property contains the value JXFS_PTR_WRITE_MICR). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

### isMSFWriteSupported

| | |
|---|---|
| **Syntax** | ***boolean isMSFWriteSupported() throws JxfsException;*** |
| **Description** | Returns TRUE if the printer has the capability to write text to the media (the *writeFormCapability* property contains the value JXFS_PTR_WRITE_MSF). |
| **Parameter** | **None** |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

# 6 Status Classes

If a device status changes one of the status classes is used in the ***StatusEvent***. A *xxxStatus* instance is passed as the *details* property of the ***StatusEvent***. Each *xxxStatus* class provides several methods to query the changed device status.
Status objects are also defined as properties in corresponding interfaces. The application has the possibility to query those properties in order to retrieve the status value it is interested in.

| Interface | Property | Description |
|---|---|---|
| *IJxfsPrinter* | ptrStatus | Contains the base device status and status objects common to all printers (toner, media and container bin) |
| *IJxfsEject* | inkStatus | Specifies the stamping ink cartridge status. |
| | exitEntryStatus | Specifies the printer's exit slot status. |
| | stackerStatus | Specifies the printer's stacker status. |
| *IJxfsRetract* | inkStatus | Specifies the stamping ink cartridge status. |
| | exitEntryStatus | Specifies the printer's exit slot status. |
| | retractBinStatus | Specifies the printer's retract bin status. |
| *IJxfsRead* | lampStatus | Specifies the status of the scanner's imaging lamp. |

### Summary

| Class | Description |
|---|---|
| JxfsMediaStatus | Used for the printing media status. |
| JxfsPtrExitEntryStatus | Used for the status of the printer's exit / entry slot. |
| JxfsPtrLampStatus | Used for the scanner's imaging lamp status. |
| JxfsPtrStatus | Container of states common to all printers. (toner, media and container bin). |
| JxfsThresholdStatus | Used for toner, container bin, stacker and retract bin. |

## 6.2 JxfsMediaStatus

This class specifies the status of the printer media. For the description of the class and its properties and methods see "Base Architecture Guide" document.

## 6.3 JxfsPtrExitEntryStatus

This class specifies the status of the printer's exit / entry slot. Only one of the flags may be true at the time. If the printer doesn't have the capability to read the exit / entry slot status, the JXFS_PTR_EXIT_ENTRY_UNKNOWN status should be used.

### 6.3.1 Summary

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

| Property | Type | Access |
|---|---|---|
| exitEntryStatus | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrExitEntryStatus | exitEntryStatus | int |

| Method | Return |
|---|---|
| isMediaAvail | *boolean* |
| isEmpty | *boolean* |
| isUnknown | *boolean* |

### 6.3.2 Properties

#### exitEntryStatus (R)

**Type** ***int***
**Initial Value** see Values below
**Description** Specifies the status of the printer imaging lamp. Depending on device capability, *lampStatus* will be set to one of the following values:

| Value | Meaning |
|---|---|
| JXFS_S_PTR_EXEN_MEDIA_AVAIL | There is media available in the exit / entry slot. |
| JXFS_S_PTR_EXEN_EMPTY | The exit / entry slot is empty. |
| JXFS_S_PTR_EXEN_UNKNOWN | State of the exit / entry slot cannot be determined with the printer in the current state. |

### 6.3.3 Methods

#### isMediaAvail

| | |
|---|---|
| **Syntax** | ***boolean isFull() throws JxfsException*** |
| **Description** | Returns TRUE if there is media available in the exit / entry slot of the device (the value of the *exitEntryStatus* property is JXFS_S_PTR_EXEN_MEDIA_AVAIL). |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

#### isEmpty

| | |
|---|---|
| **Syntax** | ***boolean isEmpty() throws JxfsException*** |
| **Description** | Returns TRUE if the exit / entry slot is empty (the value of the *exitEntryStatus* property is JXFS_S_PTR_EXEN_EMPTY). |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

#### isUnknown

| | |
|---|---|
| **Syntax** | ***boolean isUnknown() throws JxfsException*** |
| **Description** | Returns TRUE if the exit / entry slot status can not be determined with the printer in the current state or if the printer doesn't have the capability to determine the exit / entry slot status (the value of the *exitEntryStatus* property is JXFS_S_PTR_EXEN_UNKNOWN). |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

## 6.4 JxfsPtrLampStatus

This class specifies the status of the scanner's imaging lamp. Only one of the flags may be true at the time. If the printer doesn't have the capability to read the lamp status, the JXFS_S_PTR_LAMP_UNKNOWN status should be used.

### 6.4.1 Summary

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

| Property | Type | Access |
|---|---|---|
| lampStatus | int | R |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrLampStatus | lampStatus | int |

| Method | Return |
|---|---|
| isLampFading | *boolean* |
| isLampInoperable | *boolean* |
| isLampOk | *boolean* |
| isLampUnknown | *boolean* |
| isLampNotSupported | *boolean* |

### 6.4.2 Properties

#### lampStatus (R)

**Type** ***int***
**Initial Value** see Values below
**Description** Specifies the status of the printer imaging lamp. Depending on device capability, *lampStatus* will be set to one of the following values:

| Value | Meaning |
|---|---|
| JXFS_S_PTR_LAMP_OK | Imaging lamp is ok. |
| JXFS_S_PTR_LAMP_FADING | Imaging lamp should be changed. |
| JXFS_S_PTR_LAMP_INOP | Imaging lamp is inoperable. |
| JXFS_S_PTR_LAMP_UNKNOWN | State of the imaging lamp cannot be determined. |

### 6.4.3 Methods

#### isLampFading

| | |
|---|---|
| **Syntax** | ***boolean isLampFading() throws JxfsException;*** |
| **Description** | Returns TRUE if the imaging lamp should be changed (the value of the *lampStatus* property is JXFS_S_PTR_LAMP_FADING). |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

#### isLampInoperable

| | |
|---|---|
| **Syntax** | ***boolean isLampInoperable() throws JxfsException;*** |
| **Description** | Returns TRUE if the imaging lamp is inoperable (the value of the *lampStatus* property is JXFS_S_PTR_LAMP_INOP). |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

#### isLampOk

| | |
|---|---|
| **Syntax** | ***boolean isLampOk() throws JxfsException;*** |
| **Description** | Returns TRUE if the imaging lamp is ok (the value of the *lampStatus* property is JXFS_S_PTR_LAMP_OK). |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

#### isLampUnknown

| | |
|---|---|
| **Syntax** | ***boolean isLampUnknown() throws JxfsException;*** |
| **Description** | Returns TRUE if the current imaging lamp status is unknown or the printer device doesn't have the capability to read it (the value of the *lampStatus* property is JXFS_S_PTR_LAMP_UNKNOWN). |
| **Exceptions** | No additional exceptions are generated. |
| **Event** | No additional events are generated. |

#### isLampNotSupported

This method is deprecated. It is mentioned here for compatibility reasons only. The return value is always *false*.

## 6.5 JxfsPtrStatus

This class is a container of states common to all J/XFS printer devices.

### 6.5.1 Summary

**Implements :** *Serializable* **Extends :** *JxfsStatus*

| Property | Type | Access |
|---|---|---|
| mediaStatus | **JxfsMediaStatus** | R/W |
| paperStatus | **JxfsThresholdStatus** | R/W |
| tonerStatus | **JxfsThresholdStatus** | R/W |

| Constructor | Parameter | Parameter-Type |
|---|---|---|
| JxfsPtrStatus | mediaStatus | JxfsMediaStatus |
| | paperStatus | JxfsThresholdStatus |
| | tonerStatus | JxfsThresholdStatus |

| Method | Return |
|---|---|
| get*Property* | *Property* |
| set*Property* | void |

### 6.5.2 Properties

**mediaStatus (R/W)**

**Type** ***JxfsMediaStatus***
**Description** Specifies the state of the print media (i.e., the paper, passbook, single sheet, etc.).

**paperStatus (R/W)**

**Type** ***JxfsThresholdStatus***
**Description** Specifies the state of the paper supply.

**tonerStatus (R/W)**

**Type** ***JxfsThresholdStatus***
**Description** Specifies the status of the toner supply.

## 6.6 JxfsThresholdStatus

This class is used for threshold states of the following printer components: toner supply, paper supply, stacker and retract bin. Either one or none of the flags may be true at any one time, resulting with 6 possible states in total: “full”, “high”, "ok", “low”, “empty” and “unknown”. The "ok" state means that none of the flags is set. If the printer isn’t able to determine the component’s status, the “unknown” status will be reported.

If a printer can not determine some of the states defined in this class, those states won’t be reported by the device service implementation. As an example, let us assume that there are only two sensors within the paper bin: one for the critically low state and one for the empty state. Consequently, the *paperStatus* property will only be able to have 4 states: “empty”, “low”, "ok" and “unknown”, because "high" and “full” states can not be determined by this particular device.

Whenever the *JxfsThresholdStatus* object changes, a corresponding *StatusEvent* is sent to all registered listeners.

For the description of this class and its properties and methods see "Base Architecture Guide" document.

# 7 Constants

## 7.1 Alignment Codes

The alignment codes are returned by the *getAlignment()* method of the *JxfsPtrFormsConfig* class or are an input parameter for the *setAlignment()* method and the constructor of this class. The class is used to configure the usage of forms.

| Value | Meaning |
|---|---|
| JXFS_PTR_ALN_BOTTOMLEFT | Align the form to the bottom left of the media. |
| JXFS_PTR_ALN_BOTTOMRIGHT | Align the form to the bottom right of the media. |
| JXFS_PTR_ALN_TOPLEFT | Align the form to the top left of the media. |
| JXFS_PTR_ALN_TOPRIGHT | Align the form to the top right of the media. |

## 7.2 Base Unit Codes

The base unit codes are returned by the *getBase()* method of the *JxfsPtrFormsConfig* class or are an input parameter for the *setBase()* method and the constructor of this class. The class is used to configure the usage of forms.

| Value | Meaning |
|---|---|
| JXFS_PTR_FRM_INCH | Base unit is inches. |
| JXFS_PTR_FRM_MM | Base unit is millimeters. |
| JXFS_PTR_FRM_ROWCOLUMN | Base unit is rows and columns. |

## 7.3 Capability Codes

### Eject Status Capability Codes

The eject status capability codes are returned by the *getEjectStatusCapability()* method of the class *JxfsPtrEjectStatusCapability*. The values can be or'ed.

| Value | Meaning |
|---|---|
| JXFS_PTR _STATUS_INK | Device can determine the stamping ink cartridge status. |
| JXFS_PTR_STATUS_EXITENTRY | Device can determine the exit / entry slot status. |
| JXFS_PTR_STATUS_STACKER | Device can determine the stacker status. |

### Extent Capability Codes

The extent capability codes are returned by the *getExtentCapability()* method of the class *JxfsPtrExtentCapability*. The values can be or'ed.

| Value | Meaning |
|---|---|
| JXFS_PTR_EXT_HORIZONTAL | Device has horizontal size detection capability. |
| JXFS_PTR_EXT_VERTICAL | Device has vertical size detection capability. |

### Printer Status Capability Codes

The printer status capability codes are returned by the *getStatusCapability()* method of the class *JxfsPtrStatusCapability*. The values can be or'ed.

| Value | Meaning |
|---|---|
| JXFS_PTR_STATUS_TONER | Printer can determine the toner status. |
| JXFS_PTR_STATUS_PAPER | Printer can determine the paper status. |
| JXFS_PTR_STATUS_MEDIA | Printer can determine the media status. |

### Read Form Capability Codes

The read form capability codes are returned by the *getReadFormCapability()* method of the class *JxfsPtrReadFormCapability*. The values can be or'ed.

| Value | Meaning |
|---|---|
| JXFS_PTR_READ_BARCODE | Device has Barcode capability. |
| JXFS_PTR_READ_IMAGE | Device has imaging capability. |
| JXFS_PTR_READ_MICR | Device has MICR capability. |
| JXFS_PTR_READ_MSF | Device has MSF capability. |
| JXFS_PTR_READ_OCR | Device has OCR capability. |
| JXFS_PTR_READ_TEXT | Device has Text capability. |
| JXFS_PTR_READ_PAGEMARK | Device has pagemark capability. |

### Read Image Capability Codes

The read image capability codes are returned by the *getReadImageCapability()* method of the class *JxfsPtrReadImageCapability*. The values can be or'ed.

| Value | Meaning |
|---|---|
| JXFS_PTR_IMAGE_TIF | Device has capability to read tif. |
| JXFS_PTR_IMAGE_MTF | Device has capability to read mtf format. |
| JXFS_PTR_IMAGE_BMP | Device has capability to read bmp format. |

### Read Status Capability Codes

The read status capability codes are returned by the *getReadStatusCapability()* method of the class *JxfsPtrReadStatusCapability*. The values can be or'ed.

| Value | Meaning |
|---|---|
| JXFS_PTR_STATUS_LAMP | Device has the capability to determine the scanner's imaging lamp status. |

### Write Capability Codes

Following write capability codes can be or'ed.

| Value | Meaning |
|---|---|
| JXFS_PTR_WRITE_BARCODE | Device has Barcode capability. |
| JXFS_PTR_WRITE_GRAPHICS | Device has Graphics capability. |
| JXFS_PTR_WRITE_MICR | Device has MICR capability. |
| JXFS_PTR_WRITE_MSF | Device has MSF capability. |
| JXFS_PTR_WRITE_OCR | Device has OCR capability. |
| JXFS_PTR_WRITE_TEXT | Device has Text capability. |

## 7.4 Control Media Codes

The control media codes are returned by the ctrlMediaCapability method or are an input parameter for the ctrlMedia method. The codes can be or'ed.

| Value | Meaning |
|---|---|
| JXFS_PTR_CTRL_ALARM | Device can / should ring a bell, beep or otherwise sound an audible alarm. |
| JXFS_PTR_CTRL_STAMP | Device can / should stamp the media. |
| JXFS_PTR_CTRL_CUT | Device can / should cut the media. |
| JXFS_PTR_CTRL_EJECT | Device can / should eject the media. |
| JXFS_PTR_CTRL_FLUSH | Internal data buffer should be cleared and all data stored in it should be sent to the printer device immediately. |
| JXFS_PTR_CTRL_PARTIALCUT | Device can / should partially cut the media. Cut media can be easily ripped off by the |

| | |
|---|---|
| | customer. |
| JXFS_PTR_CTRL_PERFORATE | Device can / should perforate the media. Perforated media is harder to rip off than the one which was partially cut. |
| JXFS_PTR_CTRL_RETRACT | Device can / should retract the media. |
| JXFS_PTR_CTRL_SKIP | Device can / should skip to the next print mark. |
| JXFS_PTR_CTRL_STACK | Device can / should stack media items before ejecting as a bundle. |

## 7.5 Control Turn Media Codes

The control turn media codes are returned by the ctrlTurnCapability method.

| Value | Meaning |
|---|---|
| JXFS_PTR_CTRL_ATPBACKWARD | Device can/shall turn one page backward. |
| JXFS_PTR_CTRL_ATP_FORWARD | Device can/shall turn one page forward. |
| JXFS_PTR_CTRL_TURNMEDIA | Device can/shall turn the media. |

## 7.6 Error Codes

### Operation Complete Error Codes

These codes are used in *OperationCompleteEvent* events as results in order to indicate that the operation wasn't completed successfully.

| Value | Meaning |
|---|---|
| JXFS_E_PTR_EXIT_ENTRY_FAILURE | A failure occurred while ejecting / retracting the media. |
| JXFS_E_PTR_FIELD_ERROR | An error occurred while processing a field, causing termination of the print request. Details can be found in the *extendedErrorCode*. |
| JXFS_E_PTR_FIELD_NOT_FOUND | Specified field does not exist. |
| JXFS_E_PTR_FIELD_SPEC_FAILURE | Syntax of the *fieldWriteData* is invalid. |
| JXFS_E_PTR_FLUSH_FAIL | Printer was not able to flush data. |
| JXFS_E_PTR_FORM_INVALID | Specified form definition is invalid |
| JXFS_E_PTR_FORM_NOT_FOUND | Specified form definition cannot be found. |
| JXFS_E_PTR_INK_EMPTY | The stamping ink cartridge is empty. |
| JXFS_E_PTR_MEDIA_JAM | The printing media is jammed. |
| JXFS_E_PTR_MEDIA_INVALID | Specified media definition is invalid. |
| JXFS_E_PTR_MEDIA_NOT_FOUND | Specified media definition cannot be found. |
| JXFS_E_PTR_MEDIA_OVERFLOW | Form overflowed the media. |
| JXFS_E_PTR_MEDIA_SKEWED | Media skew exceeded the limit in the form definition. |
| JXFS_E_PTR_MEDIA_TURN_FAIL | Printer was not able to turn the inserted media |
| JXFS_E_PTR_NOFORMS | There are no form descriptions available on the printer. |
| JXFS_E_PTR_NO_MEDIA_PRESENT | Media is not present in the printer. |
| JXFS_E_PTR_NOMEDIA | There are no media descriptions available on the printer. |
| JXFS_E_PTR_PAPEROUT | The printer has run out of paper while printing data. Some data could have been printed. |
| JXFS_E_PTR_RETRACT_BIN_FULL | Retract bin is full. No more media can be retracted. Current media is still in the printer's exit / entry slot. Note that some printers can not distinguish this case from the JXFS_MEDIA_JAM error. |

| | |
|---|---|
| JXFS_E_PTR_STACKER_FULL | Stacker is full. No more media can be stacked. Current media is still in the print position. Note that some printers can not distinguish this case from the JXFS_MEDIA_JAM error. |
| JXFS_E_PTR_TONER_EMPTY | The printer's toner cartridge is empty. |

#### Field Failure Error Codes

These error codes are used in the *JxfsFieldFailure* object in order to report the kind of the failure.

| Value | Meaning |
|---|---|
| JXFS_E_PTR_FIELD_GRAPHIC | Specified graphic image could not be printed (the *printForm()* method) or read (the *readForm()* method). |
| JXFS_E_PTR_FIELD_HW_ERROR | Specified field uses special hardware and an error occurred. |
| JXFS_E_PTR_FIELD_NOT_READ | Attempt was made to read an output field. |
| JXFS_E_PTR_FIELD_NOT_WRITE | Attempt was made to write to an input field. |
| JXFS_E_PTR_FIELD_OVERFLOW | Value specified for the field is too long |
| JXFS_E_PTR_FIELD_REQUIRED | Specified field *must* be supplied by the application. |
| JXFS_E_PTR_FIELD_STATIC_OVWR | Specified field is *static* and thus cannot be overwritten by the application. |
| JXFS_E_PTR_FIELD_TYPE_NOT_ SUPPORTED | Form field type is not supported by the printer. |

## 7.7 Exception Codes

| Value | Meaning |
|---|---|
| JXFS_E_CLOSED | Device has not been opened yet. |
| JXFS_E_NOT_SUPPORTED | Operation is not supported by device. |
| JXFS_E_PARAMETER_INVALID | An invalid parameter was given to the operation. |
| JXFS_E_REMOTE | Communication error during remote call. |
| JXFS_E_UNREGISTERED | The Device Control object has not been registered yet |

## 7.8 Intermediate event codes

| Value | Meaning |
|---|---|
| JXFS_I_PTR_NO_MEDIA_PRESENT | No print media to print on. |
| JXFS_I_PTR_MEDIA_INSERTED | Print media has been inserted. |
| JXFS_I_PTR_FIELD_FAILURE | A failure occurred while printing or reading a form. |

## 7.9 Operation ID Codes

Following codes specify the operation which generated the OperationCompleteEvent.

| Value | Method |
|---|---|
| JXFS_O_PTR_ATP_BACKWARD | *atpBackward* |
| JXFS_O_PTR_ATP_FORWARD | *atpForward* |
| JXFS_O_PTR_CTRL_MEDIA | *ctrlMedia* |
| JXFS_O_PTR_EJECT_MEDIA | *ejectMedia* |
| JXFS_O_PTR_FIELD_INFO | *getFieldDescription* |
| JXFS_O_PTR_FORM_INFO | *getFormDescription* |
| JXFS_O_PTR_FORM_LIST | *getFormList* |
| JXFS_O_PTR_MEDIA_EXTENTS | *mediaExtents* |

| | |
|---|---|
| JXFS_O_PTR_MEDIA_INFO | *getMediaDescription* |
| JXFS_O_PTR_MEDIA_LIST | *getMediaList* |
| JXFS_O_PTR_PREPARE_EJECT | *prepareEject* |
| JXFS_O_PTR_RESET_PRINTER | *resetPrinter* |
| JXFS_O_PTR_READ_FORM_DATA | *readForm* |
| JXFS_O_PTR_READ_IMAGE | *readImage* |
| JXFS_O_PTR_RETRACT_MEDIA | *retractMedia* |
| JXFS_O_PTR_TURN_MEDIA | *turnMedia* |
| JXFS_O_PTR_WRITE_FORM_DATA | *printForm* |
| JXFS_O_PTR_WRITE_RAW_DATA | *printRawData* |

## 7.10 Status Codes

### Bin Status Codes

Defines the status code the paper supply, the toner supply or the retain bin can report.

| Value | Meaning |
|---|---|
| JXFS_S_BIN_EMPTY | Bin is empty. |
| JXFS_S_BIN_FULL | Bin is full. |
| JXFS_S_BIN_HIGH | Bin is high. |
| JXFS_S_BIN_LOW | Bin is low. |
| JXFS_S_BIN_UNKNOWN | State of the bin is unknown. |

### Exit / Entry Slot Status Codes

Exit / entry slot status codes define the status the exit / entry slot can report.

| Value | Meaning |
|---|---|
| JXFS_S_PTR_EXEN_MEDIA_AVAIL | There is media in the exit / entry slot. |
| JXFS_S_PTR_EXEN_EMPTY | The exit / entry slot is empty. |
| JXFS_S_PTR_EXEN_UNKNOWN | The exit / entry slot status is unknown. |

### General Status Codes

General Status Codes that specify a status change of the one of printer's components.

| Value | Meaning |
|---|---|
| JXFS_S_PTR_EXIT_ENTRY | The exit / entry slot status has changed. |
| JXFS_S_PTR_LAMP | The scanner's imaging lamp status has changed. |
| JXFS_S_PTR_MEDIA | The media status has changed. |
| JXFS_S_PTR_PAPER | The paper status has changed. |
| JXFS_S_PTR_RETRACT_BIN | The retract bin status has changed. |
| JXFS_S_PTR_RETRACTCOUNT | The retract count has changed. |
| JXFS_S_PTR_STACKER | The stacker status has changed. |
| JXFS_S_PTR_STACKERCOUNT | The stacker count has changed. |
| JXFS_S_PTR_TONER | The toner status has changed. |

### Lamp Status Codes

Defines the status the scanner's imaging lamp can report.

| Value | Meaning |
|---|---|
| JXFS_S_PTR_LAMP_FADING | Imaging lamp should be changed. |
| JXFS_S_PTR_LAMP_INOP | Imaging lamp is inoperable. |
| JXFS_S_PTR_LAMP_OK | Imaging lamp is ok. |
| JXFS_S_PTR_LAMP_UNKNOWN | State of the imaging lamp is unknown. |

### Media Status Codes

Defines the status codes that can be reported for the media.

| Value | Meaning |
|---|---|
| JXFS_S_MEDIA_JAMMED | Media is jammed in the device. |
| JXFS_S_MEDIA_NOTPRESENT | Media is currently not in the print position nor on the stacker. |
| JXFS_S_MEDIA_PRESENT | Media is currently in the print position or on the stacker. |
| JXFS_S_MEDIA_UNKNOWN | State of the media is unknown. |

# 8 Device Service Interface Methods

There are 5 device service interfaces which inherit from the *IJxfsBaseService*. They are: *IJxfsPrinterService*, *IJxfsEjectService*, *IJxfsMediaTurnService*, *IJxfsReadService* and *IJxfsRetractService*.

The Device Service interface is common to all device services of this device type. It is used by the Device Controls to access the functionality of the device. This interface has to be implemented by any J/XFS Device Service. The device type specific Device Service interface is similar to the Device Control interface. All device specific method calls are extended by an additional parameter (int control_id). This is always added as the last parameter in every operation.

# 9 Form, Field and Media Definitions

For the definition of forms, the fields within them, and the media on which they are printed see the XFS specification: „Version 2.0, CWA 13449-3:1998“.

# 10 APPENDIX A : CEN/ISSS WORKSHOP 14923:2004 CORE MEMBERS :

| | |
|---|---|
| **DELARUE** | |
| **DIEBOLD** |  |
| **DYNASTY** |  |
| **IBM** |  |
| **KAL** | |
| **KEBA** | |
| **LUTZ WOLF GRUPPE** |  |
| **NCR** |  |
| **NEXUS** | |
| **SEIKO EPSON CORPORATION** | |
| **WINCOR - NIXDORF** |  |

**< End of Document >**