CEN

WORKSHOP

AGREEMENT

CWA 16008-6

August 2009

ICS 35.240.40

English version

J/eXtensions for Financial Services (J/XFS) for the Java Platform - Release 2009 - Part 6: Printer Device Class Interface - Programmer's Reference

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

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

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

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

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

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

Management Centre: Avenue Marnix 17, B-1000 Brussels

Ref. No.:CWA 16008-6:2009 E

Contents

FOREWORD **5**

1 HISTORY **7**

2 SCOPE **8**

3 OVERVIEW **9**
3.1 DESCRIPTION 9
3.2 CLASS HIERARCHY 10
3.3 CLASS AND INTERFACE SUMMARY 11

4 DEVICE BEHAVIOR **13**
4.1 BASE SERVICE BEHAVIOR 13
4.2 HANDLING OF *NULL* PARAMETERS 13
4.3 PAPER VS. MEDIA 13
4.4 EXIT / ENTRY SLOT 13

5 CLASSES AND INTERFACES **14**
5.1 ACCESS TO PROPERTIES 14
5.2 EXCEPTIONS 14
5.3 IJXFSPRINTERCONTROL 15
5.3.1 Summary 15
5.3.2 Properties 16
5.3.3 Methods 16
5.4 IJXFSEJECT 24
5.4.1 Summary 24
5.4.2 Properties 24
5.4.3 Methods 25
5.5 IJXFSRETRACT 28
5.5.1 Summary 28
5.5.2 Properties 28
5.5.3 Methods 28
5.6 IJXFSMEDIATURN 30
5.6.1 Summary 30
5.6.2 Properties 30
5.6.3 Methods 30
5.7 IJXFSREAD 32
5.7.1 Summary 32
5.7.2 Properties 32
5.7.3 Methods 32

6 SUPPORT CLASSES **37**
6.2 JXFSPTRCTRLMEDIACAPABILITY 38
6.2.1 Summary 38
6.2.2 Properties 38
6.2.3 Methods 38
6.3 JXFSPTRCTRLTURNCAPABILITY 40
6.3.1 Summary 40
6.3.2 Properties 40
6.3.3 Methods 40
6.4 JXFSPTREJECTSTATUSCAPABILITY 41
6.4.1 Summary 41
6.4.2 Properties 41
6.4.3 Methods 41
6.5 JXFSPTREXTENTCAPABILITY 42
6.5.1 Summary 42
6.5.2 Properties 42
6.5.3 Methods 42
6.6 JXFSPTRFIELD 43
6.6.1 Summary 43
6.6.2 Properties 43
6.7 JXFSPTRFIELDFAILURE 45

6.7.1 Summary 45
6.7.2 Properties 45
6.8 JXFSPTRFORM 46
6.8.1 Summary 46
6.8.2 Properties 47
6.9 JXFSPTRFORMSCONFIG 49
6.9.1 Summary 49
6.9.2 Properties 49
6.10 JXFSPTRIMAGE 51
6.10.1 Summary 51
6.10.2 Properties 51
6.11 JXFSPTRMAXRETRACTCAPABILITY 52
6.11.1 Summary 52
6.11.2 Properties 52
6.12 JXFSPTRMAXSTACKERCAPABILITY 53
6.12.1 Summary 53
6.12.2 Properties 53
6.13 JXFSPTRMEDIA 54
6.13.1 Summary 54
6.13.2 Properties 54
6.14 JXFSPTRMEDIAEXTENTS 58
6.14.1 Summary 58
6.14.2 Properties 58
6.15 JXFSPTRREADFORMCAPABILITY 59
6.15.1 Summary 59
6.15.2 Properties 59
6.15.3 Methods 59
6.16 JXFSPTRREADIMAGECAPABILITY 61
6.16.1 Summary 61
6.16.2 Properties 61
6.16.3 Methods 61
6.17 JXFSPTRREADSTATUSCAPABILITY 62
6.17.1 Summary 62
6.17.2 Properties 62
6.17.3 Methods 62
6.18 JXFSPTRSTATUSCAPABILITY 63
6.18.1 Summary 63
6.18.2 Properties 63
6.18.3 Methods 63
6.19 JXFSPTRRETRACTCOUNT 64
6.19.1 Summary 64
6.19.2 Properties 64
6.19.3 Methods 64
6.20 JXFSPTRSTACKERCOUNT 65
6.20.1 Summary 65
6.20.2 Properties 65
6.20.3 Methods 65
6.21 JXFSPTRWRITEFORMCAPABILITY 66
6.21.1 Summary 66
6.21.2 Properties 66
6.21.3 Methods 66
6.22 JXFSPTRCAPABILITIES 67
6.22.1 Properties 67
6.22.2 Constructors 67

7 STATUS CLASSES 68
7.2 JXFSMEDIASTATUS 69
7.3 JXFSPTREXITENTRYSTATUS 70
7.3.1 Summary 70
7.3.2 Properties 70
7.3.3 Methods 71
7.4 JXFSPTRLAMPSTATUS 72

7.4.1 Summary ... 72
7.4.2 Properties ... 72
7.4.3 Methods ... 73
7.5 JXFSPTRSTATUS ... 74
7.5.1 Summary ... 74
7.5.2 Properties ... 74
7.5.3 Constructors ... 75
7.6 JXFSTHRESHOLDSTATUS ... 76

8 ENUM CLASSES ... 77
8.1 JXFSPTRPAPERSOURCEENUM ... 77
8.2 JXFSPTRSTATUSSELECTORENUM ... 77

9 CONSTANTS ... 78
9.1 ALIGNMENT CODES ... 78
9.2 BASE UNIT CODES ... 78
9.3 CAPABILITY CODES ... 78
9.4 CONTROL MEDIA CODES ... 79
9.5 CONTROL TURN MEDIA CODES ... 80
9.6 ERROR CODES ... 80
9.7 FORMS AND MEDIA CODES ... 82
9.7.1 Form Configuration Offset Codes ... 82
9.7.2 Form Orientation Codes ... 82
9.7.3 Field Access Mode Codes ... 82
9.7.4 Field Class Codes ... 82
9.7.5 Field Type Codes ... 82
9.7.6 Field Data Overflow Codes ... 83
9.7.7 Media Type ... 83
9.7.8 Media Fold Type ... 83
9.8 INTERMEDIATE EVENT CODES ... 83
9.9 OPERATION ID CODES ... 83
9.10 STATUS CODES ... 84

10 DEVICE SERVICE INTERFACE METHODS ... 85

11 FORM, FIELD AND MEDIA DEFINITIONS ... 86

12 CLARIFICATIONS OF FORMS AND MEDIA AMBIGUITIES ... 87
12.1 FORMS DEFINITION ... 87
12.1.1 General behavior ... 87
12.1.2 Form attributes ... 88
12.1.3 Field attributes ... 88
12.1.4 Frame attributes ... 91
12.2 MEDIA DEFINITION ... 92
12.2.1 General behavior ... 92
12.2.2 Attributes ... 92

Foreword

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

The CEN J/XFS Workshop gathers suppliers (among others the J/XFS Forum members), service providers as well as banks and other financial service companies. A list of companies participating in this Workshop and in support of this CWA is available from the CEN Secretariat, and at http://www.cen.eu/cenorm/sectors/sectors/isss/activity/jxfs_membership.asp. The specification was agreed upon by the J/XFS Workshop Meeting of 2009-05 -6/9 in Brussels, and the final version was sent to CEN for publication on 2009-06-12.

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

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

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

This CWA is composed of the following parts:

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

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

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

1 History

Main differences to CWA 14923-6:2004 are:

- Clarification 247 on exitEntryStatus
- Numerical values for constants specified
- Forms and media ambiguities clarification added
- Replaced OperationCompleteEvent by JxfsOperationCompleteEvent and also for IntermediateEvent and StatusEvent
- Support for multiple paper sources

Main differences to CWA 13937-6:2000 are:

- descriptions of *JxfsMediaStatus* and *JxfsThresholdStatus* classes made conform to the "Base Architecture" document
- *IJxfsRetract* interface extends the *IJxfsEject* interface
- included description of handling *null* parameters
- 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 is 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:JxfsPtrEjectStatusCapabilit, 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.

2 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.

3 Overview

3.1 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.

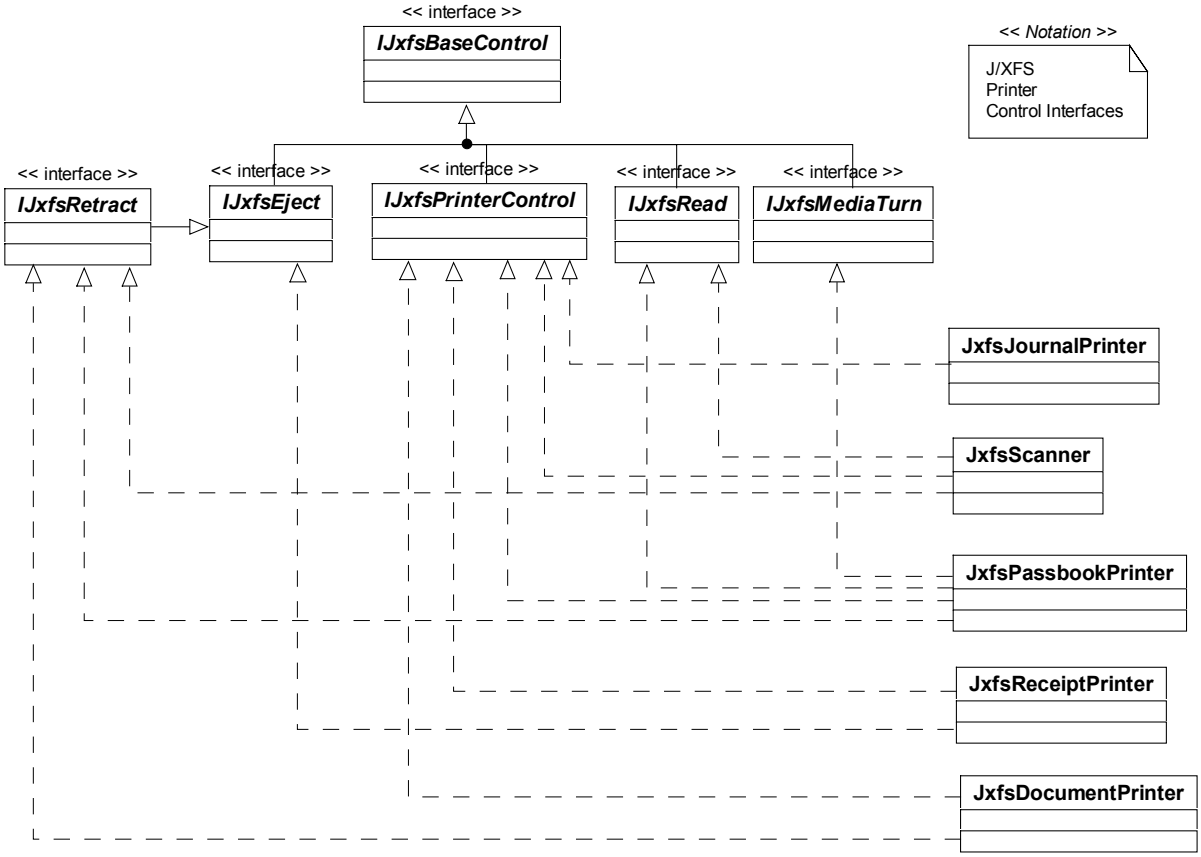
- **Passbook Printer**
  The passbook device is physically and functionally the most complex printer. The J/XFS definition supports automatic positioning of the book, as well as read/write capability for an optional integrated magnetic stripe. The implementation also manages the geometry of the book - 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 calls 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.

3.2 Class Hierarchy



3.3 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.	--

4 Device behavior

4.1 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.

4.2 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.

4.3 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.

4.4 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.

5 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, a *JxfsOperationCompleteEvent* is generated which contains detailed information about the status of the operation, i.e. if it failed or succeeded, and eventually additional data as a result.

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

5.1 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. JXFS_E_CLOSED JXFS_E_REMOTE 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. JXFS_E_CLOSED JXFS_E_PARAMETER_INVALID JXFS_E_REMOTE JXFS_E_UNREGISTERED

5.2 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 codes can also appear as the *result* value within *JxfsOperationCompleteEvent* events. Only if a method can throw an exception with an additional error code or send a *JxfsOperationCompleteEvent* event with a different result, it is explicitly mentioned in this document.

5.3 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.

5.3.1 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
ptrCapabilities	**JxfsPtrCapabilities**	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
setCurrentPaperSource	identificationID

Event	May occur during / after
JxfsStatusEvent	
JXFS_S_PTR_MEDIA	*ctrlMedia(), mediaExtents(),printForm(), printRawData(), resetPrinter()*
JXFS_S_PTR_PAPER	*printForm(), printRawData(), resetPrinter()*
JXFS_S_PTR_TONER	*printForm(), printRawData(), resetPrinter()*
JxfsIntermediateEvent	
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
JxfsOperationCompleteEvent	
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	*mediaExtents()*
JXFS_O_PTR_WRITE_FORM_DATA	*printForm()*
JXFS_O_PTR_WRITE_RAW_DATA	*printRawData()*
JXFS_O_PTR_RESET_PRINTER	*resetPrinter()*

5.3.2 Properties

compound (R)

This property is deprecated. It is mentioned here for compatibility reasons only. Its 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 its 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 *JxfsStatusEvent* to all registered listeners.

writeFormCapability (R)

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

ptrCapabilities (R)

Type ***JxfsPtrCapabilities***
Initial Value see *JxfsPtrCapabilities*
Description This property specifies printer capabilities for multiple paper sources.

5.3.3 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 values: JXFS_PTR_CTRL_ALARM JXFS_PTR_CTRL_FLUSH JXFS_PTR_CTRL_SKIP For descriptions of those flags see chapter 9: "Constants".

Events Following events can be generated:

JxfsOperationCompleteEvent

When a *ctrlMedia()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	none

JxfsStatusEvent

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

Field	Value
status	JXFS_S_PTR_MEDIA
details	JxfsMediaStatus mediaStatus The new printer media status.

getFieldDescription

Syntax ***identificationID getFieldDescription(java.lang.java.lang.String[] fieldNames, java.lang.java.lang.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
java.lang.java.lang.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. The array is not allowed to contain *null* entries.
java.lang.String	formName	Name of the requested form.

Events Following events can be generated:

JxfsOperationCompleteEvent

When a *getFieldDescription()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).

data	*JxfsPtrField[]* aFieldDefs 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(java.lang.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
java.lang.String	formName	Name of the requested form.

Events Following events can be generated:

JxfsOperationCompleteEvent

When a *getFormDescription()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	*JxfsPtrForm* aJxfsPtrForm 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**

Events Following events can be generated:

JxfsOperationCompleteEvent

When a *getFormList()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	java.lang.String[] aFormsList 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(java.lang.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
java.lang.String	mediaName	Name of the requested media.

Events Following events can be generated:

JxfsOperationCompleteEvent

When a *getMediaDescription()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	*JxfsPtrMedia* aJxfsPtrMedia 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**

Events Following events can be generated:

JxfsOperationCompleteEvent

When a *getMediaList()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	java.lang.String[] aMediaList 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**

Events Following events can be generated:

JxfsOperationCompleteEvent

When a *mediaExtents()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	*JxfsPtrMediaExtents* aJxfsPtrMediaExtents The extents of the inserted media. If an error occurs this field will be *null* as no media extents could be returned.

JxfsIntermediateEvent

If no media is present the J/XFS Printer Device Control will send a *JxfsIntermediateEvent* 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

JxfsIntermediateEvent
If media is inserted and the operation can continue the J/XFS Printer Device Control will send a *JxfsIntermediateEvent* 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

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

Field	**Value**
status	JXFS_S_PTR_MEDIA
details	JxfsMediaStatus mediaStatus The new printer media status.

printForm

Syntax

identificationID printForm(java.lang.String formName, java.lang.String mediaName, java.lang.String[] fieldWriteData) throws JxfsException;

Description

This method prints the form with the name *formName* using the description of the media defined by *mediaName.* The paper source to be used is determined by *IJxfsPrinterControl.ptrStatus.currentPaperSource*. After a successful completion of this output operation, a *JxfsOperationCompleteEvent* 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 a *JxfsOperationCompleteEvent* 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**
java.lang.String	formName	Name of the form to be printed.
java.lang.String	mediaName	Name of the media to be used for printing.
java.lang.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". This array is not allowed to contain *null* entries.

Events

Following events can be generated:

JxfsOperationCompleteEvent
When a *printForm()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	none

JxfsIntermediateEvent
If no media is present the J/XFS Printer Device Control will send a *JxfsIntermediateEvent* 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

JxfsIntermediateEvent
If media is inserted and the operation can continue the J/XFS Printer Device Control will send a *JxfsIntermediateEvent* 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

JxfsIntermediateEvent
If a field error occurs during printing the field and the Device Service is capable to continue with printing the further fields[1], a *JxfsIntermediateEvent* 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 More detailed information about the failure.

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

Field	**Value**
status	JXFS_S_PTR_PAPER
details	*JxfsThresholdStatus* paperStatus The new paper supply status.

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

Field	**Value**
status	JXFS_S_PTR_TONER
details	*JxfsThresholdStatus* tonerStatus The new toner supply status.

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

Field	**Value**
status	JXFS_S_PTR_MEDIA
details	JxfsMediaStatus media The new printer media status.

printRawData

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

[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.

Description

This command is used to send raw data (a byte string of device dependent data) to the physical device. The paper source to be used is determined by *IJxfsPrinterControl.ptrStatus.currentPaperSource*. 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 *JxfsOperationCompleteEvent* is initialized properly.

Printers with paper source (e.g. journal and receipt printers) will send a *JxfsOperationCompleteEvent* 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 its state objects. This could cause an unpredictable behavior. For the same reason, various error codes can be returned as the *result* field of the *JxfsOperationCompleteEvent* 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.

Events

Following events can be generated:

JxfsOperationCompleteEvent

When a *printRawData()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	byte[] inputData 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.

JxfsIntermediateEvent

If no media is present the J/XFS Printer Device Control will send a *JxfsIntermediateEvent* 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

JxfsIntermediateEvent

If media is inserted and the operation can continue the J/XFS Printer Device Control will send a *JxfsIntermediateEvent* 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

JxfsStatusEvent
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 its 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**

Events Following events can be generated:

JxfsOperationCompleteEvent
When a *resetPrinter()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	none

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

setCurrentPaperSource

Syntax ***identificationID setCurrentPaperSource(JxfsPtrPaperSourceEnum currentPaperSource) throws JxfsException;***

Description Sets the current paper source.
Not all printers support switching the paper supply in all situations. Some printers may not allow this operation during an active printing job or if the paper size of the old and new supply differs too much. Therefore it is recommended to issue this command only when there is no media processed.

Parameter

Type	Name	Description
JxfsPtrPaperSourceEnum	currentPaperSource	Specifies the paper source to be set.

Events Following events can be generated:

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

operationID	JXFS_O_PTR_SET_CURRENT_PAPER_SOURCE
identificationID	The corresponding ID
result	Common or device dependent error code. (See section on *Error Codes*).
data	none

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

5.4 IJxfsEject

5.4.1 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()*
JxfsOperationCompleteEvent	
JXFS_O_PTR_EJECT_MEDIA	*ejectMedia()*
JXFS_O_PTR_PREPARE_EJECT	*prepareEject()*

5.4.2 Properties

ejectStatusCapability (R)

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

exitEntryStatus (R)

Type ***JxfsPtrExitEntryStatus***
Initial Value see JxfsPtrExitEntryStatus
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 ***JxfsPtrStackerCount***
Initial Value see JxfsPtrStackerCount
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.

5.4.3 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 values: JXFS_PTR_CTRL_ALARM JXFS_PTR_CTRL_FLUSH JXFS_PTR_CTRL_SKIP JXFS_PTR_CTRL_CUT JXFS_PTR_CTRL_PARTIALCUT JXFS_PTR_CTRL_PERFORATE JXFS_PTR_CTRL_STACK JXFS_PTR_CTRL_STAMP For descriptions of those flags see chapter 9: "Constants".

Events Following events can be generated:

JxfsOperationCompleteEvent
When an *ejectMedia()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	none

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

Field	Value
status	JXFS_S_PTR_MEDIA
details	*JxfsMediaStatus* mediaStatus The new printer media status.

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

Field	Value
status	JXFS_S_PTR_EXIT_ENTRY
details	*JxfsPtrExitEntryStatus* exitEntryStatus The new printer exit slot status.

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

Field	Value
status	JXFS_S_PTR_INK
details	*JxfsThresholdStatus* inkStatus The new printer stamp ink cartridge status.

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

Field	Value
status	JXFS_S_PTR_STACKER

details	*JxfsThresholdStatus* stackerStatus The new stacker status.	

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

Field	**Value**
status	JXFS_S_PTR_STACKERCOUNT
details	*JxfsPtrStackerCount* stackerCount 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 a *JxfsOperationCompleteEvent* 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 values: JXFS_PTR_CTRL_ALARM JXFS_PTR_CTRL_FLUSH JXFS_PTR_CTRL_SKIP JXFS_PTR_CTRL_CUT JXFS_PTR_CTRL_PARTIALCUT JXFS_PTR_CTRL_PERFORATE JXFS_PTR_CTRL_STACK JXFS_PTR_CTRL_STAMP For descriptions of those flags see chapter 9: "Constants".

Events
Following events can be generated:
JxfsOperationCompleteEvent
When a *prepareEject()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	none

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

Field	**Value**
status	JXFS_S_PTR_MEDIA
details	*JxfsMediaStatus* mediaStatus The new media status.

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

Field	**Value**
status	JXFS_S_PTR_INK

details	*JxfsThresholdStatus* inkStatus The new printer stamping ink cartridge status.

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

Field	**Value**
status	JXFS_S_PTR_STACKER
details	*JxfsThresholdStatus* stackerStatus The new stacker status.

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

Field	**Value**
status	JXFS_S_PTR_STACKERCOUNT
details	*JxfsPtrStackerCount* stackerCount The new stacker counter value.

5.5 IJxfsRetract

5.5.1 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()*
JxfsOperationCompleteEvent	
JXFS_O_PTR_RETRACT_MEDIA	*retractMedia()*

5.5.2 Properties

maxRetractCapability (R)

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

5.5.3 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 values: JXFS_PTR_CTRL_ALARM JXFS_PTR_CTRL_FLUSH JXFS_PTR_CTRL_CUT JXFS_PTR_CTRL_STAMP For descriptions of those flags see chapter 9: "Constants".

Events

Following events can be generated:

JxfsOperationCompleteEvent

When a *retractMedia()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	none

JxfsStatusEvent

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

Field	**Value**
status	JXFS_S_PTR_MEDIA
details	*JxfsMediaStatus* mediaStatus The new printer media status.

JxfsStatusEvent

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

Field	**Value**
status	JXFS_S_PTR_RETRACT_BIN
details	*JxfsThresholdStatus* retractBinStatus The new retract bin status.

JxfsStatusEvent

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

Field	**Value**
status	JXFS_S_PTR_RETRACTCOUNT
details	*JxfsPtrRetractCount* retractCount The new retract counter value.

JxfsStatusEvent

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

Field	**Value**
status	JXFS_S_PTR_EXIT_ENTRY
details	*JxfsThresholdStatus* exitSlotStatus The new printer exit slot status.

JxfsStatusEvent

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

Field	**Value**
status	JXFS_S_PTR_INK
details	*JxfsThresholdStatus* inkStatus The new printer stamp ink cartridge status.

5.6 IJxfsMediaTurn

5.6.1 Summary

Property	Type	Access
ctrlTurnCapability	**JxfsPtrCtrlTurnCapability**	R

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

Event	May occur during / after
JxfsStatusEvent JXFS_S_PTR_MEDIA	 *atpBackward(), atpForward(), turnMedia()*
JxfsOperationCompleteEvent JXFS_O_PTR_ATP_BACKWARD JXFS_O_PTR_ATP_FORWARD JXFS_O_PTR_TURN_MEDIA	 *atpBackward()* *atpForward()* *turnMedia()*

5.6.2 Properties

ctrlTurnCapability (R)

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

5.6.3 Methods

atpBackward

Syntax ***identificationID atpBackward() throws JxfsException;***
Description This command is used to turn the page of the passbook backward.
Parameter **none**
Events Following events can be generated:

JxfsOperationCompleteEvent
When a *atpBackward()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	none

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

Field	Value
status	JXFS_S_PTR_MEDIA
details	*JxfsMediaStatus* mediaStatus 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**
Events Following events can be generated:
JxfsOperationCompleteEvent

When a *atpForward()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	none

JxfsStatusEvent

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

Field	**Value**
status	JXFS_S_PTR_MEDIA
details	*JxfsMediaStatus* mediaStatus The new media status.

turnMedia

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

Description This command is used to turn the inserted media.

Parameter **none**

Events Following events can be generated:

JxfsOperationCompleteEvent

When a *turnMedia()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	none

JxfsStatusEvent

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

Field	**Value**
status	JXFS_S_PTR_MEDIA
details	*JxfsMediaStatus* mediaStatus The new media status.

5.7 IJxfsRead

5.7.1 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()*
JxfsOperationCompleteEvent	
JXFS_O_PTR_READ_FORM_DATA	*readForm()*
JXFS_O_PTR_READ_IMAGE	*readImage()*

5.7.2 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.

5.7.3 Methods

readForm

Syntax ***identificationID readForm(java.lang.String formName, java.lang.String mediaName, java.lang.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.* The paper source to be used is determined by *IJxfsPrinterControl.ptrStatus.currentPaperSource*. After a successful completion of this input operation, a *JxfsOperationCompleteEvent* 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
java.lang.String	formName	Name of the form to be read.
java.lang.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 *JxfsOperationCompleteEvent*.
java.lang.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.

Events

Following events can be generated:

JxfsOperationCompleteEvent

When a *readForm()* operation is completed a *JxfsOperationCompleteEvent* 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	Common or device dependent error code. (See section on *Error Codes*).
data	java.lang.String [] readData 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.

JxfsIntermediateEvent

If no media is present the J/XFS Printer Device Control will send a *JxfsIntermediateEvent* 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

JxfsIntermediateEvent

If media is inserted and the operation can continue the J/XFS Printer Device Control will send a *JxfsIntermediateEvent* 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

JxfsIntermediateEvent
If a field error occurs during reading the field data and the Device Service is capable to continue with reading the further fields[2], a *JxfsIntermediateEvent* 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 More detailed information about the failure.

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

Field	**Value**
status	JXFS_S_PTR_MEDIA
details	*JxfsMediaStatus* mediaStatus The new media status.

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

Field	**Value**
status	JXFS_S_PTR_LAMP
details	*JxfsPtrLampStatus* lampStatus 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(java.lang.String formName, java.lang.String mediaName, java.lang.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.* The paper source to be used is determined by *IJxfsPrinterControl.ptrStatus.currentPaperSource*. After a successful completion of this input operation, a *JxfsOperationCompleteEvent* 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**
java.lang.String	formName	Name of the form to be read.
java.lang.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 *JxfsOperationCompleteEvent*.
java.lang.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.

Events

Following events can be generated:

JxfsOperationCompleteEvent

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

identificationID	The corresponding ID
result	Common or device dependent error code. (See section on *Error Codes*).
data	*JxfsPtrImage[]* readData An array of images successfully read by this operation.

JxfsIntermediateEvent

If no media is present the J/XFS Printer Device Control will send a *JxfsIntermediateEvent* 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

JxfsIntermediateEvent

If media is inserted and the operation can continue the J/XFS Printer Device Control will send a *JxfsIntermediateEvent* 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

JxfsIntermediateEvent

If a field error occurs during reading the image data and the Device Service is capable to continue with reading the further fields[3], a *JxfsIntermediateEvent* 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 More detailed information about the failure.

JxfsStatusEvent

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

Field	**Value**
status	JXFS_S_PTR_MEDIA
details	*JxfsMediaStatus* mediaStatus The new media status.

JxfsStatusEvent

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

Field	**Value**
status	JXFS_S_PTR_LAMP
details	*JxfsPtrLampStatus* lampStatus The new lamp status.

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

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.

6 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 its 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 *printForm()*, *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 its 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 its components.
JxfsPtrRetractCount	Specifies the retract counter.
JxfsPtrWriteFormCapability	Specifies write form capabilities.
JxfsPtrCapabilities	Specifies paper sources.

6.2 JxfsPtrCtrlMediaCapability

This class specifies the control media capabilities of the printer.

6.2.1 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

6.2.2 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

6.2.3 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**

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**

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**

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**

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**

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**

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**

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**

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**

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**

6.3 JxfsPtrCtrlTurnCapability

This class specifies the turn media capabilities of the printer.

6.3.1 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

6.3.2 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

6.3.3 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**

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**

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**

6.4 JxfsPtrEjectStatusCapability

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

6.4.1 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

6.4.2 Properties

ejectStatusCapability (R)

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

6.4.3 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**

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**

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**

6.5 JxfsPtrExtentCapability

This class specifies the extent capability of the printer.

6.5.1 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

6.5.2 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.

6.5.3 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**

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**

6.6 JxfsPtrField

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

6.6.1 Summary

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

Property	Type	Access
access	int	R
classType	int	R
fieldName	java.lang.String	R
format	java.lang.String	R
indexCount	int	R
initialValue	java.lang.String	R
overflow	int	R
type	int	R

Constructor	Parameter	Parameter-Type
JxfsPtrField	access	int
	classType	int
	fieldName	java.lang.String
	format	java.lang.String
	indexCount	int
	initialValue	java.lang.String
	overflow	int
	type	int

Method	Return
get*Property*	*Property*

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

format (R)

Type *java.lang.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 *java.lang.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.

6.7 JxfsPtrFieldFailure

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

6.7.1 Summary

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

Property	Type	Access
fieldFailure	int	R
fieldName	java.lang.String	R
formName	java.lang.String	R

Constructor	Parameter	Parameter-Type
JxfsPtrFieldFailure	fieldFailure	int
	fieldName	java.lang.String
	formName	java.lang.String

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

6.7.2 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 ***java.lang.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 ***java.lang.String***
Initial Value empty String
Description Specifies the name of the form at which the error occurred.

6.8 JxfsPtrForm

The JxfsPtrForm class contains the properties of a specified form.

6.8.1 Summary

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

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

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

Method	Return
get*Property*	*Property*

6.8.2 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 ***java.lang.String[]***
Initial Value empty String[]
Description Indicates the field names on the form

formName (R)

Type ***java.lang.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.

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.

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_LAND SCAPE	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 ***java.lang.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.

6.9 JxfsPtrFormsConfig

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

6.9.1 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 #1	Parameter	Parameter-Type
JxfsPtrFormsConfig	*none*	

Constructor #2	Parameter	Parameter-Type
JxfsPtrFormsConfig	alignment	int
	base	int
	offsetX	int
	offsetY	int
	unitX	int
	unitY	int

Method	Return
get*Property*	*Property*
set*Property*	void

6.9.2 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_FRM_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. 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. A value of JXFS_PTR_ FRM_OFFSET_USEFORMDEFN specifies that the *offsetX* from the form definition should be used.

offsetY (R/W)

Type	***int***
Initial Value	JXFS_PTR_ FRM_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. 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. This value is specified in terms of the unitY property and is always positive. A value of JXFS_PTR_ FRM_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.

6.10 JxfsPtrImage

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

6.10.1 Summary

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

Property	Type	Access
fieldName	java.lang.String	R
imageData	byte[]	R
imageType	int	R

Constructor	Parameter	Parameter-Type
JxfsPtrImage	fieldName	java.lang.String
	imageData	byte[]
	imageType	int

Method	Return
get*Property*	*Property*

6.10.2 Properties

fieldName (R)

Type ***java.lang.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.

6.11 JxfsPtrMaxRetractCapability

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

6.11.1 Summary

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

Property	Type	Access
maxRetractCapability	int	R

Constructor	Parameter	Parameter-Type
JxfsPtrMaxRetractCapability	maxRetractCapability	int

Method	Return
get*Property*	*Property*

6.11.2 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).

6.12 JxfsPtrMaxStackerCapability

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

6.12.1 Summary

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

Property	Type	Access
maxStackerCapability	int	R

Constructor	Parameter	Parameter-Type
JxfsPtrMaxStackerCapability	maxStackerCapability	int

Method	Return
get*Property*	*Property*

6.12.2 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).

6.13 JxfsPtrMedia

The JxfsPtrMedia class contains the properties of a specified media.

6.13.1 Summary

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

Property	Type	Access
base	int	R
foldtype	int	R
lineCount	int	R
mediaName	java.lang.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	java.lang.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*

6.13.2 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 *java.lang.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.

6.14 JxfsPtrMediaExtents

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

6.14.1 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*

6.14.2 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.

6.15 JxfsPtrReadFormCapability

This class specifies the read form capabilities of the printer.

6.15.1 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

6.15.2 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.

6.15.3 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**

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**

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**

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**

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**

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**

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**

6.16 JxfsPtrReadImageCapability

This class specifies the read image capabilities of the printer.

6.16.1 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

6.16.2 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.

6.16.3 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**

isImageMTFSupported

Syntax ***boolean isImageMTFSupported() throws JxfsException;***
Description Returns TRUE if the device has the capability to read mtf format (the *readImageCapability* property contains the value JXFS_PTR_IMAGE_MTF).
Parameter **None**

isImageBMPSupported

Syntax ***boolean isImageBMPSupported() throws JxfsException;***
Description Returns TRUE if the device has the capability to read bmp format (the *readImageCapability* property contains the value JXFS_PTR_IMAGE_BMP).
Parameter **None**

6.17 JxfsPtrReadStatusCapability

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

6.17.1 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

6.17.2 Properties

readStatusCapability (R)

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

6.17.3 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**

6.18 JxfsPtrStatusCapability

This class specifies the capabilities of the printer to determine states of its components.

6.18.1 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

6.18.2 Properties

statusCapability (R)

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

6.18.3 Methods

isTonerStatusSupported

Syntax ***boolean isTonerStatusSupported() throws JxfsException;***
Description Returns TRUE if the printer has the capability to determine the toner status.
Parameter **None**

isMediaStatusSupported

Syntax ***boolean isTonerStatusSupported() throws JxfsException;***
Description Returns TRUE if the printer has the capability to determine the media status.
Parameter **None**

isPaperStatusSupported

Syntax ***boolean isTonerStatusSupported() throws JxfsException;***
Description Returns TRUE if the printer has the capability to determine the paper bin status.
Parameter **None**

6.19 JxfsPtrRetractCount

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

6.19.1 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

6.19.2 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.

6.19.3 Methods

resetRetractCount

Syntax ***void resetRetractCount() throws JxfsException;***
Description Sets the number of retracts to zero.
Parameter **None**

6.20 JxfsPtrStackerCount

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

6.20.1 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

6.20.2 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.

6.20.3 Methods

resetStackerCount

Syntax ***void resetStackerCount() throws JxfsException***
Description Sets the number of stacked media to zero.
Parameter **None**

6.21 JxfsPtrWriteFormCapability

This class specifies the write form capabilities of the printer.

6.21.1 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

6.21.2 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.

6.21.3 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**

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**

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**

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**

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**

isMSFWriteSupported

Syntax ***boolean isMSFWriteSupported() throws JxfsException;***
Description Returns TRUE if the printer has the capability to write MSF to the media (the *writeFormCapability* property contains the value JXFS_PTR_WRITE_MSF).
Parameter **None**

6.22 JxfsPtrCapabilities

Extends	Implements
JxfsType	

Property	Type	Access
supportedPaperSources	*JxfsPtrPaperSourceEnum[]*	R

Constructor	Parameter	Parameter-Type
JxfsPtrCapabilities	SupportedPaperSources	*JxfsPtrPaperSourceEnum[]*

Method	Return
get*Property*	*Property*

6.22.1 Properties

supportedPaperSources (R)

Type ***JxfsPtrPaperSourceEnum[]***
Remarks Returns an array containing one entry for each paper source available to the printer. An empty array indicates that the actual configuration is not known.

6.22.2 Constructors

JxfsPtrCapabilities

Syntax ***public JxfsPtrCapabilities(JxfsPtrPaperSourceEnum[] supportedPaperSources) throws JxfsException***
Remarks If the device service does not support several paper sources the supportedPaperSources object must hold an 'any' enum value only.
Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID	Generated if supportedPaperSources is a null reference.

7 Status Classes

If a device status changes one of the status classes is used in the *JxfsStatusEvent*. An *xxxStatus* instance is passed as the *details* property of the *JxfsStatusEvent*. 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
IJxfsPrinterControl	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.

7.2 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.

7.3 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.

7.3.1 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*

7.3.2 Properties

exitEntryStatus (R)

Type ***int***
Initial Value see Values below
Description Specifies the status of the printer's *exit / entry slot*. Depending on device capability, *exitEntryStatus* 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.

7.3.3 Methods

isMediaAvail

Syntax ***boolean isMediaAvail() 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).

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).

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).

7.4 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.

7.4.1 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*

7.4.2 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.

7.4.3 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).

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).

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).

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).

isLampNotSupported

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

7.5 JxfsPtrStatus

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

7.5.1 Summary

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

Property	Type	Access
mediaStatus	JxfsMediaStatus	R
paperStatus	JxfsThresholdStatus	R
tonerStatus	JxfsThresholdStatus	R
paperSourceStatus	java.util.Map	R
currentPaperSource	JxfsPtrPaperSourceEnum	R

Constructor #1	Parameter	Parameter-Type
JxfsPtrStatus	mediaStatus	JxfsMediaStatus
	paperStatus	JxfsThresholdStatus
	tonerStatus	JxfsThresholdStatus

Constructor #2	Parameter	Parameter-Type
JxfsPtrStatus	mediaStatus	JxfsMediaStatus
	paperStatus	JxfsThresholdStatus
	tonerStatus	JxfsThresholdStatus
	paperSourceStatus	java.util.Map
	currentPaperSource	JxfsPtrPaperSourceEnum

Method	Return
get*Property*	*Property*
set*Property*	void

7.5.2 Properties

mediaStatus (R)

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

paperStatus (R)

Type ***JxfsThresholdStatus***
Description Specifies the threshold status of the selected JxfsPtrPaperSourceEnum paper source.

tonerStatus (R)

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

paperSourceStatus (R)

Type ***java.util.Map***
Description Specifies the threshold status of the available paper sources.
The key/value pair types are as shown below:
keys: *JxfsPtrPaperSourceEnum* objects
values: *JxfsThresholdStatus* objects

currentPaperSource (R)

Type ***JxfsPtrPaperSourceEnum***
Description Specifies the paper source currently selected.

7.5.3 Constructors

JxfsPtrStatus

Syntax ***public JxfsPtrStatus(JxfsMediaStatus mediaStatus, JxfsThresholdStatus paperStatus, JxfsThresholdStatus tonerStatus)***

Remarks paperSourceStatus will be set to a java.util.Map with one object containing 'any' as key and the paperStatus parameter as value.
currentPaperSource will be set to 'any'.

JxfsPtrStatus

Syntax ***public JxfsPtrStatus(JxfsMediaStatus mediaStatus, JxfsThresholdStatus tonerStatus, java.util.Map paperSourceStatus, JxfsPtrPaperSourceEnum currentPaperSource) throws JxfsException***

Remarks The constructor sets the paperStatus property to the element of paperSourceStatus with key currentPaperSource.

Exceptions Exceptions, which can be generated by this method.

JXFS_E_PARAMETER_INVALID	Generated if one of the following cases applies: - keys of paperSourceStatus Map are not of the type *JxfsPtrPaperSourceEnum* - values of paperSourceStatus Map are not of the type *JxfsThresholdStatus* - currentPaperSource is not included in paperSourceStatus.

7.6 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 *JxfsStatusEvent* is sent to all registered listeners.

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

8 Enum Classes

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

8.1 JxfsPtrPaperSourceEnum

Extends	Implements
JxfsEnum	

Field	Description
any	Default source. The printer may choose its paper source automatically. If currentPaperSource is equal to 'any' then the paperStatus of *JxfsPtrStatus* is the combined status of all possible paper sources.
single	Identifies the printer single paper source.
upper	Identifies the printer upper paper source.
lower	Identifies the printer lower paper source.
aux	Identifies the printer auxiliary paper source.
aux2	Identifies the printer second auxiliary paper source.
external	Identifies the printer external supply (e.g. envelope tray or single sheet feed).

8.2 JxfsPtrStatusSelectorEnum

Extends	Implements
JxfsStatusSelectorEnum	

Field	Returned Type	Description
ptrStatus	*JxfsPtrStatus*	Status of the printer device.
exitEntryStatus	*JxfsPtrExitEntryStatus*	Status of the printer's exit / entry slot. This status is available only if the device service implements the eject interface.
inkStatus	*JxfsThresholdStatus*	Status of the stamping ink cartridge. This status is available only if the device service implements the eject interface.
stackerCount	*JxfsPtrStackerCount*	Number of stacked medias. This count is available only if the device service implementes the eject interface.
stackerStatus	*JxfsThresholdStatus*	Status of the printer's stacker. This status is available only if the device service implements the eject interface.
retractBinStatus	*JxfsThresholdStatus*	Status of the printer's retract bin. This status is available only if the device service implementes the retract interface.
retractCount	*JxfsPtrRetractCount*	Number of retracted medias. This count is available only if the device service implements the retract interface.
lampStatus	*JxfsPtrLampStatus*	Status of the scanner's imaging lamp status. This status is available only if the device service implements the read interface.

9 Constants

9.1 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.

Code	Value	Meaning
3107	JXFS_PTR_ALN_BOTTOMLEFT	Align the form to the bottom left of the media.
3108	JXFS_PTR_ALN_BOTTOMRIGHT	Align the form to the bottom right of the media.
3105	JXFS_PTR_ALN_TOPLEFT	Align the form to the top left of the media.
3106	JXFS_PTR_ALN_TOPRIGHT	Align the form to the top right of the media.
3109	JXFS_PTR_ALN_USEFORMDEFN	Use alignment specified in the form definition.

9.2 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.

Code	Value	Meaning
3102	JXFS_PTR_FRM_INCH	Base unit is inches.
3103	JXFS_PTR_FRM_MM	Base unit is millimeters.
3104	JXFS_PTR_FRM_ROWCOLUMN	Base unit is rows and columns.

9.3 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.

Code	Value	Meaning
1	JXFS_PTR _STATUS_INK	Device can determine the stamping ink cartridge status.
2	JXFS_PTR_STATUS_EXIT_ENTRY	Device can determine the exit / entry slot status.
4	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.

Code	Value	Meaning
1024	JXFS_PTR_EXT_HORIZONTAL	Device has horizontal size detection capability.
2048	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.

Code	Value	Meaning
1	JXFS_PTR_STATUS_TONER	Printer can determine the toner status.
2	JXFS_PTR_STATUS_PAPER	Printer can determine the paper status.
4	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.

Code	Value	Meaning
4096	JXFS_PTR_READ_BARCODE	Device has Barcode capability.
8192	JXFS_PTR_READ_IMAGE	Device has imaging capability.
16384	JXFS_PTR_READ_MICR	Device has MICR capability.
32768	JXFS_PTR_READ_MSF	Device has MSF capability.
65536	JXFS_PTR_READ_OCR	Device has OCR capability.
131072	JXFS_PTR_READ_TEXT	Device has Text capability.
262144	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*.

Code	Value	Meaning
3112	JXFS_PTR_IMAGE_TIF	Device has capability to read tif.
3113	JXFS_PTR_IMAGE_MTF	Device has capability to read mtf format.
3114	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 *JxfsPtrStatusCapability*. The values can be or'ed.

Code	Value	Meaning
1	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.

Code	Value	Meaning
524288	JXFS_PTR_WRITE_BARCODE	Device has Barcode capability.
1048576	JXFS_PTR_WRITE_GRAPHICS	Device has Graphics capability.
2097152	JXFS_PTR_WRITE_MICR	Device has MICR capability.
4194304	JXFS_PTR_WRITE_MSF	Device has MSF capability.
8388608	JXFS_PTR_WRITE_OCR	Device has OCR capability.
16777216	JXFS_PTR_WRITE_TEXT	Device has Text capability.

9.4 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.

Code	Value	Meaning
1	JXFS_PTR_CTRL_ALARM	Device can / should ring a bell, beep or otherwise sound an audible alarm.
2	JXFS_PTR_CTRL_STAMP	Device can / should stamp the media.
4	JXFS_PTR_CTRL_CUT	Device can / should cut the media.
8	JXFS_PTR_CTRL_EJECT	Device can / should eject the media.
16	JXFS_PTR_CTRL_FLUSH	Internal data buffer should be cleared and all data stored in it should be sent to the printer device immediately.
32	JXFS_PTR_CTRL_PARTIALCUT	Device can / should partially cut the media. Cut media can be easily ripped off by the customer.
64	JXFS_PTR_CTRL_PERFORATE	Device can / should perforate the media. Perforated media is harder to rip off than the one which was partially cut.
128	JXFS_PTR_CTRL_RETRACT	Device can / should retract the media.
256	JXFS_PTR_CTRL_SKIP	Device can / should skip to the next print mark.
512	JXFS_PTR_CTRL_STACK	Device can / should stack media items before ejecting as a bundle.

9.5 Control Turn Media Codes

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

Code	Value	Meaning
33554432	JXFS_PTR_CTRL_ATP_BACKWARD	Device can turn one page backward.
67108864	JXFS_PTR_CTRL_ATP_FORWARD	Device can turn one page forward.
134217728	JXFS_PTR_CTRL_TURNMEDIA	Device can turn the media.

9.6 Error Codes

Error and Exception Codes

These codes are used in *JxfsOperationCompleteEvent* events as results or they are thrown in the synchroneous part of a method in order to indicate that the operation wasn't completed successfully.

Code	Value	Meaning
3060	JXFS_E_PTR_EXIT_ENTRY_FAILURE	A failure occurred while ejecting / retracting the media.
3001	JXFS_E_PTR_EXTEND_NOT_ SUPPORTED	Measuring media extents is not supported by the printer.
3002	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*.
3003	JXFS_E_PTR_FIELD_INVALID	The specified field is invalid.
3004	JXFS_E_PTR_FIELD_NOT_FOUND	Specified field does not exist.
3005	JXFS_E_PTR_FIELD_SPEC_FAILURE	Syntax of the *fieldWriteData* is invalid.

3006	JXFS_E_PTR_FLUSH_FAIL	Printer was not able to flush data.
3007	JXFS_E_PTR_FORM_INVALID	Specified form definition is invalid
3010	JXFS_E_PTR_FORM_NOT_FOUND	Specified form definition cannot be found.
3061	JXFS_E_PTR_INK_EMPTY	The stamping ink cartridge is empty.
3062	JXFS_E_PTR_MEDIA_JAM	The printing media is jammed.
3011	JXFS_E_PTR_MEDIA_INVALID	Specified media definition is invalid.
3012	JXFS_E_PTR_MEDIA_NOT_FOUND	Specified media definition cannot be found.
3013	JXFS_E_PTR_MEDIA_OVERFLOW	Form overflowed the media.
3014	JXFS_E_PTR_MEDIA_SKEWED	Media skew exceeded the limit in the form definition.
3015	JXFS_E_PTR_MEDIA_TURN_FAIL	Printer was not able to turn the inserted media
3008	JXFS_E_PTR_NOFORMS	There are no form descriptions available on the printer.
3016	JXFS_E_PTR_NO_MEDIA_PRESENT	Media is not present in the printer.
3009	JXFS_E_PTR_NOMEDIA	There are no media descriptions available on the printer.
3038	JXFS_E_PTR_PAPEROUT	The printer has run out of paper while printing data. Some data could have been printed.
3017	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.
3063	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.
3064	JXFS_E_PTR_TONER_EMPTY	The printer's toner cartridge is empty.

Field Failure Error Codes

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

Code	Value	Meaning
3018	JXFS_E_PTR_FIELD_GRAPHIC	Specified graphic image could not be printed (the *printForm()* method) or read (the *readForm()* method).
3019	JXFS_E_PTR_FIELD_HW_ERROR	Specified field uses special hardware and an error occurred.
3020	JXFS_E_PTR_FIELD_NOT_READ	Attempt was made to read an output field.
3021	JXFS_E_PTR_FIELD_NOT_WRITE	Attempt was made to write to an input field.
3022	JXFS_E_PTR_FIELD_OVERFLOW	Value specified for the field is too long.
3023	JXFS_E_PTR_FIELD_REQUIRED	Specified field *must* be supplied by the application.

Code	Value	Meaning
3024	JXFS_E_PTR_FIELD_STATIC_OVWR	Specified field is *static* and thus cannot be overwritten by the application.
3025	JXFS_E_PTR_FIELD_TYPE_NOT_ SUPPORTED	Form field type is not supported by the printer.

9.7 Forms and Media Codes

The constants listed in this chapter are used for specifying properties describing form and media definitions.

9.7.1 Form Configuration Offset Codes

These constants are used for specifying the *offsetX* and *offsetY* properties of the *JxfsPtrFormsConfig* class.

Code	Value	Meaning
2999	JXFS_PTR_FRM_OFFSET_ USEFORMDEFN	This value indicates that the value specified in the form definition is to be used.

9.7.2 Form Orientation Codes

These constants are used for specifying the *orientation* property of the *JxfsPtrForm* class.

Code	Value	Meaning
3111	JXFS_PTR_FRM_LANDSCAPE	Orientation of the form is landscape.
3110	JXFS_PTR_FRM_PORTRAIT	Orientation of the form is portrait.

9.7.3 Field Access Mode Codes

These constants are used for specifying the *access* property of the *JxfsPtrField* class. The values can be or'ed.

Code	Value	Meaning
33554432	JXFS_PTR_FRM_ACCESS_READ	Field is used for input.
67108864	JXFS_PTR_FRM_ACCESS_WRITE	Field is used for output.

9.7.4 Field Class Codes

These constants are used for specifying the *classType* property of the *JxfsPtrField* class.

Code	Value	Meaning
3087	JXFS_PTR_FRM_CLASS_OPTIONAL	Field data can be set by the application.
3089	JXFS_PTR_FRM_CLASS_REQUIRED	Field data must be set by the application.
3088	JXFS_PTR_FRM_CLASS_STATIC	Field data cannot be set by the application.

9.7.5 Field Type Codes

These constants are used for specifying the *type* property of the *JxfsPtrField* class.

Code	Value	Meaning
3095	JXFS_PTR_FRM_FIELD_BARCODE	Barcode field.
3096	JXFS_PTR_FRM_FIELD_GRAPHIC	Graphic field.
3097	JXFS_PTR_FRM_FIELD_MICR	Magnetic Ink Character Recognition field.
3098	JXFS_PTR_FRM_FIELD_MSF	Magnetic Stripe Facility field.

Code	Value	Meaning
3099	JXFS_PTR_FRM_FIELD_OCR	Optical Recognition Character field.
3100	JXFS_PTR_FRM_FIELD_PAGEMARK	Page Mark field.
3101	JXFS_PTR_FRM_FIELD_TEXT	Text field.

9.7.6 Field Data Overflow Codes

These constants are used for specifying the *overflow* property of the *JxfsPtrField* class.

Code	Value	Meaning
3092	JXFS_PTR_FRM_OVF_BEST_FIT	Fit text in the field.
3093	JXFS_PTR_FRM_OVF_OVERWRITE	Print field data beyond the extents of the field boundary.
3090	JXFS_PTR_FRM_OVF_TERMINATE	Return an error and terminate printing the form.
3091	JXFS_PTR_FRM_OVF_TRUNCATE	Truncate field data to fit in the field.
3094	JXFS_PTR_FRM_OVF_WORDWRAP	If field can hold more than one line the text is wrapped around.

9.7.7 Media Type

These constants are used for specifying the *mediaType* property of the *JxfsPtrMedia* class.

Code	Value	Meaning
3118	JXFS_PTR_FRM_MEDIA_GENERIC	Generic media, i.e., single sheet.
3119	JXFS_PTR_FRM_MEDIA_ MULTIPART	Multipart media.
3120	JXFS_PTR_FRM_MEDIA_PASSBOOK	Passbook media.

9.7.8 Media Fold Type

These constants are used for specifying the *foldType* property of the *JxfsPtrMedia* class.

Code	Value	Meaning
3115	JXFS_PTR_FRM_FOLD_ HORIZONTAL	Passbook has horizontal fold.
3117	JXFS_PTR_FRM_FOLD_NONE	Passbook has no fold.
3116	JXFS_PTR_FRM_FOLD_VERTICAL	Passbook has vertical fold.

9.8 Intermediate event codes

Code	Value	Meaning
3123	JXFS_I_PTR_NO_MEDIA_PRESENT	No print media to print on.
3121	JXFS_I_PTR_MEDIA_INSERTED	Print media has been inserted.
3140	JXFS_I_PTR_FIELD_FAILURE	A failure occurred while printing or reading a form.

9.9 Operation ID Codes

Following codes specify the operation which generated the *JxfsOperationCompleteEvent*.

Code	Value	Method
3074	JXFS_O_PTR_ATP_BACKWARD	*atpBackward*
3075	JXFS_O_PTR_ATP_FORWARD	*atpForward*
3076	JXFS_O_PTR_CTRL_MEDIA	*ctrlMedia*
3077	JXFS_O_PTR_EJECT_MEDIA	*ejectMedia*
3125	JXFS_O_PTR_FIELD_INFO	*getFieldDescription*
3126	JXFS_O_PTR_FORM_INFO	*getFormDescription*
3127	JXFS_O_PTR_FORM_LIST	*getFormList*
3078	JXFS_O_PTR_MEDIA_EXTENTS	*mediaExtents*

3128	JXFS_O_PTR_MEDIA_INFO	*getMediaDescription*
3129	JXFS_O_PTR_MEDIA_LIST	*getMediaList*
3079	JXFS_O_PTR_PREPARE_EJECT	*prepareEject*
3080	JXFS_O_PTR_RESET_PRINTER	*resetPrinter*
3081	JXFS_O_PTR_READ_FORM_DATA	*readForm*
3082	JXFS_O_PTR_READ_IMAGE	*readImage*
3083	JXFS_O_PTR_RETRACT_MEDIA	*retractMedia*
3084	JXFS_O_PTR_TURN_MEDIA	*turnMedia*
3085	JXFS_O_PTR_WRITE_FORM_DATA	*printForm*
3086	JXFS_O_PTR_WRITE_RAW_DATA	*printRawData*
3095	JXFS_O_PTR_SET_CURRENT_PAPER_SOURCE	*setCurrentPaperSource*

9.10 Status Codes

Exit / Entry Slot Status Codes

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

Code	Value	Meaning
3221	JXFS_S_PTR_EXEN_MEDIA_AVAIL	There is media in the exit / entry slot.
3220	JXFS_S_PTR_EXEN_EMPTY	The exit / entry slot is empty.
3222	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.

Code	Value	Meaning
3040	JXFS_S_PTR_EXIT_ENTRY	The exit / entry slot status has changed.
3026	JXFS_S_PTR_LAMP	The scanner's imaging lamp status has changed.
3027	JXFS_S_PTR_MEDIA	The media status has changed.
3028	JXFS_S_PTR_PAPER	The paper status has changed.
3029	JXFS_S_PTR_RETRACT_BIN	The retract bin status has changed.
3030	JXFS_S_PTR_RETRACTCOUNT	The retract count has changed.
3041	JXFS_S_PTR_STACKER	The stacker status has changed.
3042	JXFS_S_PTR_STACKERCOUNT	The stacker count has changed.
3031	JXFS_S_PTR_TONER	The toner status has changed.
3124	JXFS_S_PTR_INK	The printer stamp ink cartridge status has changed.

Lamp Status Codes

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

Code	Value	Meaning
3035	JXFS_S_PTR_LAMP_FADING	Imaging lamp should be changed.
3036	JXFS_S_PTR_LAMP_INOP	Imaging lamp is inoperable.
3032	JXFS_S_PTR_LAMP_OK	Imaging lamp is ok.
3034	JXFS_S_PTR_LAMP_UNKNOWN	State of the imaging lamp is unknown.

10 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.

11 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“.

12 Clarifications of Forms and Media Ambiguities

The purpose of this chapter is to define how to handle forms and media definitions in J/XFS Device Services in the cases where the XFS specification is not clear enough.

12.1 Forms Definition

12.1.1 General behavior

Obviously conflicting fields

When a form definition which contains obviously conflicting fields (e.g. defining text as superscript and subscript in the STYLE field) is referenced by name when one of the following methods is called:

- *getFormDescription(java.lang.String)*
- *getFieldDescription(java.lang.String[], java.lang.String)*
- *printForm(java.lang.String, java.lang.String, java.lang.String[] fields)*
- *readForm(java.lang.String)*
- *readForm (java.lang.String, java.lang.String, java.lang.String[])*
- *readImage(java.lang.String, java.lang.String, java.lang.String[])*

the requested operation will fail. For each erroneous field, a corresponding *JxfsIntermediateEvent* object with the reason field set to JXFS_I_PTR_FIELD_FAILURE will be sent.

Finally, a *JxfsOperationCompleteEvent* object with the result field set to JXFS_E_PTR_FIELD_ERROR will be sent to all registered listeners. The data field will be set null. The *extendedResult* field may be set to a vendor-specific value in order to provide some additional information about the error cause.
The specific *printForm()*, *readForm()* and *readImage()* operations will fail only if they are using a conflicting field. Not used conflicting fields are ignored.

Printing forms and flushing

This topic doesn't address ambiguities in forms and media definitions and is therefore not covered by this document. However, the behavior of the forms printing under these circumstances may be specified in a separate clarification document, if required.

Unsupported contents

In querying operations, Device Service does not filter out items from form, field or media definitions which are not supported by the printer hardware. For example, even if a given printer doesn't have the capability to read magnetic stripe, the *getFormDescription()* will return names of all JXFS_PTR_FRM_FIELD_MSF fields. Consequently, the *getFieldDescription()* method will also return the corresponding field description objects for those fields.

Retracting and flushing

Flushing and retracting behavior is not related to forms and media ambiguities and is therefore not covered by this document.

12.1.2 Form attributes

Semantics of the SKEW attribute

The SKEW attribute denotes the largest acceptable deviation of the paper orientation from its ideal orientation when printing the given form. It is the angle in degrees between the right (or left) edge of the print media and the right (or left) edge of the document feeder.

Usage of the USERPROMPT attribute

The interpretation of the USERPROMPT attribute is left to the application, which has the possibility to query its value using the *getFormDescription()* method. The value of this attribute doesn't affect the behavior of Device Services.

12.1.3 Field attributes

Definition of the ROWCOLUMN unit

This value is deprecated and should not be used in the future. If specified in the form definition, the interpretation of this value is vendor-specific.

Declared read/write field access and hardware capabilities

If the declared field access is not supported by the underlying hardware, an error will occur only if an operation is called, which is not supported by the hardware. That means, if a magnetic stripe field is declared as read/write and the printer is only capable of reading the magnetic stripe, the *readForm()* operation with the given field will be successful, whereas the *printForm()* operation containing the same field as argument will fail.
Failures are reported in the same way as already described for obviously conflicting fields.

Base for the POSITION attribute

The coordinates of the upper left corner of the form are always (0,0), regardless on whether MM, INCH or ROWCOLUMN is declared as the base resolution unit for the given form definition.

Codepages and the value of the LANGUAGE attribute

In J/XFS, all form definitions are represented in Unicode using one of the following encoding methods:

- UTF-8: Eight-bit UCS Transformation Format
- UTF-16BE: Sixteen-bit UCS Transformation Format, big-endian byte order
- UTF-16LE: Sixteen-bit UCS Transformation Format, little-endian byte order
- UTF-16: Sixteen-bit UCS Transformation Format, byte order identified by an optional byte-order mark

The usage of Unicode character set makes form definitions language-independent. However, many printers currently available on the market can print only a subset of Unicode characters at a time, as defined by the corresponding codepage. The printable subset is changed by switching between codepages. Due to possible huge performance losses, switching codepages in forms printing jobs may be unfeasible for some printers. Therefore, the codepage used by such printers may be specified in a vendor-dependent manner outside of the form definition.

The LANGUAGE attribute is a valid ISO Language Code. These codes are the lower-case, two-letter codes as defined by ISO 639. This attribute may be used by Device Service as hint for the codepage to use. This attribute has no meaning for printers which are capable

of printing the complete set of Unicode characters. Non-printable Unicode characters are rendered in the vendor-specific manner.

Here are some examples how various Device Service implementations for printers which are not able to display the complete Unicode character set may render the name of an imaginary son of a Croatian in Germany whose name is Jürgen Markušević:

- **Implementation 1:** The Device Service renders forms according to the codepage specified in the configuration repository and ignores the LANGUAGE attribute. Non-printable Unicode characters are rendered as question marks. If the specified codepage was ISO-8859-1 (Western), the name of the person would be rendered as "Jürgen Marku?evi?". For the codepage ISO-8859-2 (Central European), the name of the person would be rendered as "Jürgen Markušević".
- **Implementation 2:** The Device Service uses the LANGUAGE attribute as a hint for the codepage to use. Non-printable Unicode characters are rendered as nearest printable characters available in the used codepage. If the specified LANGUAGE was set to "de" (german) or "hr" (croatian), the name of our person would be rendered as "Jürgen Markušević", because the Device Service would choose the ISO 8859-2 (Central European) codepage in both cases. For the LANGUAGE attribute set to "en" (english), the name would be rendered as "Jurgen Markušević", because the ISO 8859-1 codepage (Western) would be chosen.

Semantics of the INITIALVALUE attribute for fields of the type GRAPHIC

The INITIALVALUE for fields of the type GRAPHIC may contain the path of the file on the local file system or the URL address of the resource containing the graphic image. Additionally to the file extension, the type of the graphic may also be determined by its MIME content type or by introspecting the resource data directly. Supported image data formats are vendor-specific.

Semantics of the FONT attribute

The FONT attribute contains the name of the font typeface (Times New Roman, Arial, Helvetica, etc.) without any modifiers (italic, bold, underscore, etc.). If the FONT attribute and the STYLE attribute conflict with each other (e.g. the fixed-space font is required and the proportional style is specified), it is a case of obviously conflicting fields which is to be handled as already described.

If a fixed-width font was chosen, the value for either the CPI or the POINTSIZE attribute may be defined in order to specify the size of the font. If both are defined and conflict with each other, it is the case of obviously conflicting fields. If none is defined, a vendor-dependent default font size is chosen.

If a proportional font was chosen, the POINTSIZE attribute may be defined in order to specify the size of the font (the value of the CPI attribute is ignored). If the POINTSIZE attribute is omitted, a vendor-dependent default font size is chosen.

LPI attribute and font appearance

The value of the LPI attribute does not affect the font appearance. If not specified, a vendor-dependent default value is chosen.

Semantics of the FOLLOWS attribute

In the following, we assume that we have two fields A and B with positioning attributes (x_a, y_a, $width_a$, $height_a$) for the field A and (x_b, y_b, $width_b$, $height_b$) for the field B. We also assume that it is declared that the field B follows the field A. The coordinate y_b is overwritten by the value $y_a + effectiveHeight_a + y_b$. The $effectiveHeight_a$ is calculated as follows:

- If $height_a$ is non-negative, $effectiveHeight_a$ equals to $height_a$.
- If $height_a$ is negative, $effectiveHeight_a$ equals to the calculated height of the text to be printed within the field A.
- The coordinate x_b remains unchanged.

If one of the fields is an indexed field, a vendor-specific handling may be provided. If the device service does not handle such cases, then they should be regarded as obviously conflicting fields.

Semantics of the FORMAT attribute

The interpretation of the FORMAT attribute is left to the application, which has the possibility to query its value using the *getFieldDescription()* method. The value of this attribute doesn't affect the behavior of Device Services.

Two fields have different values for the SIDE attributes, but they follow each other?

This case will be handled as described above for obviously conflicting fields, i.e. the whole form will be rejected.

Positioning of indexed fields

The position of the field with the index 'i' is calculated according to the following formula:

- $x_i = x + (i * x_{offset})$
- $y_i = y + (i * y_{offset})$

That means, the offset is measured from the top left corner. If offset is smaller than the size of the field, it may happen that cells with adjacent indexes overwrite each other.

Semantics of the REPEAT count

The REPEATCOUNT attribute of an index field denotes how often the field appears. That means, index values between 0 and REPEATCOUNT-1 (including boundaries) are valid for a given index field.

Semantics of the OVERFLOW attribute

If OVERWRITE is specified for the field A, the positioning of an arbitrary field B is never affected by the fact whether the content of the field A actually overflows the area defined for that field or not.

The TERMINATE value for the OVERFLOW attribute means that in the case of the field overflow in the *printForm()* operation, the following should occur:

- *JxfsIntermediateEvent* containing details about the error cause should be sent to all registered listeners. The exact content of this event is described in the printer specification.
- The *printForm()* operation should be aborted.
- *JxfsOperationCompleteEvent* event with the result field set to JXFS_E_FIELD_ERROR, the *extendedResult* field set to the vendor-specific error code and the data field set to null should be sent to all registered listeners.

This also happens if WORDWRAP or BESTFIT value is specified for the OVERFLOW attribute and, after applying the corresponding measures, the content still doesn't fit into the defined field boundaries.

Semantics of the STYLE attribute

Some printers have support for enlarged fonts, which can be switched on/off by appropriate escape sequences. If available, this feature should be utilized if the value of the STYLE attribute contains one (or more) of the flags DOUBLE, TRIPLE, QUADRUPLE, DOUBLEHIGH, TRIPLEHIGH or QUADRUPLEHIGH. The following table lists rows of mutually exclusive flags, i.e. at most one of the flags listed in a single row is allowed to be specified in the value of the STYLE attribute:

BOLD		
ITALIC		
UNDER	DOUBLEUNDER	
DOUBLE	TRIPLE	QUADRUPLE
STRIKETHROUGH	DOUBLESTRIKE	
ROTATE90	ROTATE270	UPSIDEDOWN
PROPORTIONAL		
DOUBLEHIGH	TRIPLEHIGH	QUADRUPLEHIGH
CONDENSED		
SUPERSCRIPT	SUBSCRIPT	
OVERSCORE		
LETTERQUALITY	NEARLETTERQUALITY	
OPAQUE		

Style settings don't affect the field size, defined by the SIZE attribute. The STYLE attribute affects only fields of the TEXT type and is ignored for any other field types. When two flags in the STYLE attribute contradict, it is handled in the same way as described above for obviously conflicting fields, i.e. the whole form will be rejected.

Colors used for printing

Actual printed colors depend on the capabilities of the underlying printer. On monochrome printers, the value WHITE for the COLOR attribute will be mapped to the invisible color (won't be printed), and all other values will be mapped to the actual ink color of the given printer (usually black). If a monochrome printer supports grayscales, the corresponding grayscale of a given color will be used.

Text alignment

All text alignments in a field are possible for all reasonable unit and CPI settings. For a ROWCOLUMN unit, it is not required to have "divisible" CPI / LPI settings. Text alignment is performed in the following steps:

- In the first step, the Device Service calculates the exact desired text position in the field, based on the field content, size, alignment and chosen font.
- Secondly, capabilities of the specific device are taken into account and the text position is rounded to the nearest position supported by the device.
- Finally, field overflow is handled according to the value of the OVERFLOW attribute.

12.1.4 Frame attributes

For common attributes, see clarifications for field attributes.

12.2 Media Definition

12.2.1 General behavior

Base for the POSITION attribute

The coordinates of the upper left corner of the media are always (0,0), regardless on whether MM, INCH or ROWCOLUMN is declared as the base resolution unit for the given media definition. This applies to all attributes which provide positioning information (currently PRINTAREA and RESTRICTED attributes provide such information).

12.2.2 Attributes

Semantics of the STAGGERING attribute

The STAGGERING attribute denotes the height of the staggering area, measured in vertical units from the top of the medium. The printable area may overlap the staggering area. A given field to be printed in a *printForm()* operation is allowed to be located either completely inside or completely outside of the staggering area.

Semantics of the PAGE attribute

The default value of the PAGE attribute is 0. The interpretation of this attribute is left to the application, which has the possibility to query its value using the *getMediaDescription()* method. The value of this attribute doesn't affect the behavior of Device Services.

Semantics of the LINES attribute

The default value for the LINES attribute is 0. The interpretation of this attribute is left to the application, which has the possibility to query its value using the *getMediaDescription()* method. The value of this attribute doesn't affect the behavior of Device Services.