CEN

WORKSHOP

AGREEMENT

CWA 14923-11

May 2004

ICS 35.240.40

Supersedes CWA 13937-11:2003

English version

J/eXtensions for Financial Sevices (J/XFS) for the Java Platform - Part 11: Camera Specification - Programmer's Reference

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

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

Ref. No.:CWA 14923-11:2004 E

Contents

Foreword

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

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

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

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

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

This CWA is composed of the following parts:

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

CWA 14923-11:2004 replaces CWA 13937-11:2003 and should be read in conjunction with CWA 13937:2000, which contains the previous release of the J/XFS specification

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

History

The CWA 14923:2004 is the first release of this CWA part.

1 Scope

This document describes the Camera Device Class (CAM) 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

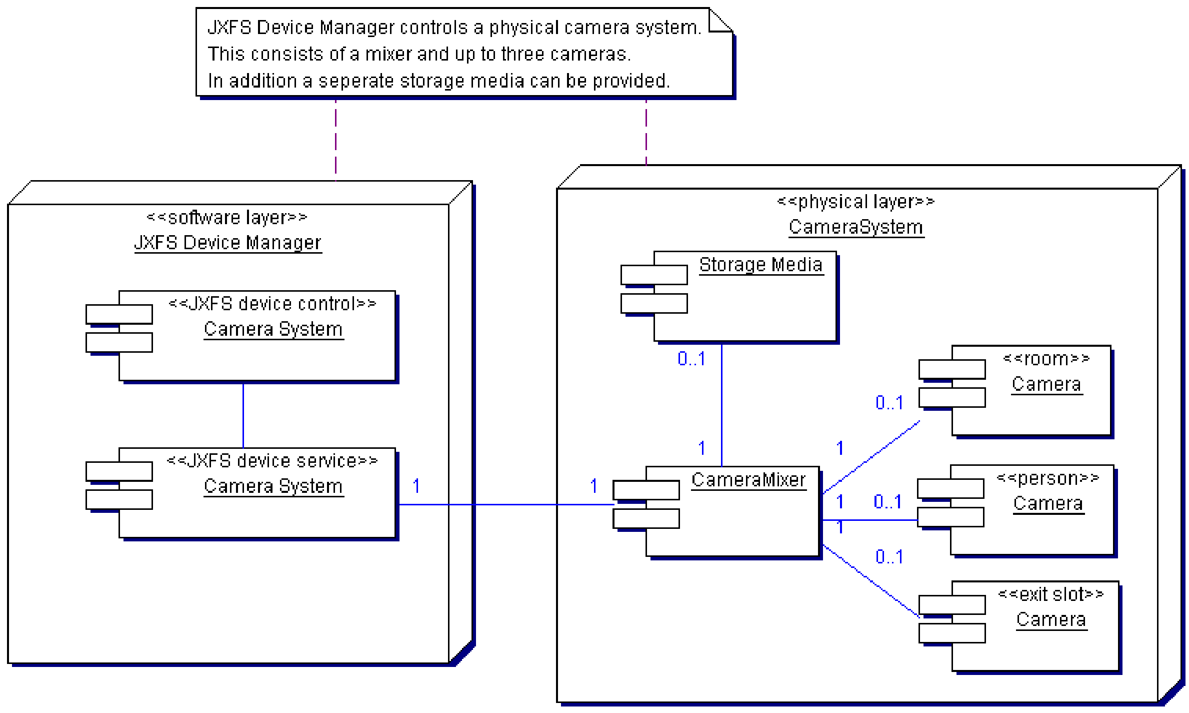
Application developers program against control objects and the Device Manager which reside in the Device Control Layer. This is the usual interface between applications and J/XFS Devices. Device Control Objects access the Device Manager to find an associated Device Service. Device Service Objects provide 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 Camera Devices, the basic Device Control structure is extended with various properties and methods specific to this device which are described on the following pages.

2 Overview

2.1 Description

This document describes the input and output features of the Camera System. It offers the functionality of a banking camera system. These camera systems usually consist of a recorder, a video mixer and one or more cameras. The diagramm shows the basic structure of a camera system inside a J/XFS-environment (standard configuration).



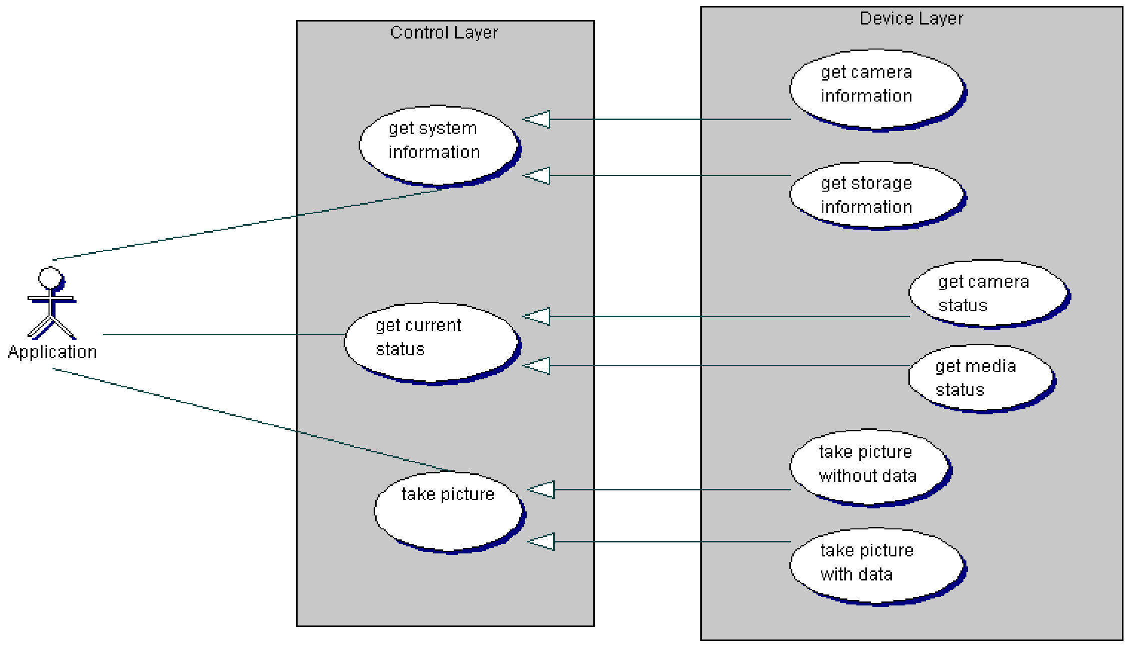
This configuration is the most common used in the field but can vary in different installations.
If there are several cameras, each camera has a focus on a special place within the self-service area. The standard configuration consists of three cameras with focus on the room, the customer or the cash tray. In addition to these standard focustypes the vendor of the camera-system can apply additional cameras with arbitrary focus.

By using the video mixer it can be decided which of the cameras should take the next photo. Furthermore data can be given to be inserted in the photo (e.g. date, time or bankcode). The picture is usually stored in a single storage media that is connected to the camera mixer.

Instead of having three (physical) cameras it is possible that only one single camera is present. In this case the camera may be able to take photos from different positions.

2.2 Description of Use-Cases

The interface includes methods to access the existing cameras in order to initiate and execute single photo shots. Any kind of streaming video is not part of this interface. The following diagramm shows the use cases that describe the main functionality of the camera system:



2.2.1 Use Case 1: Get System Information

In this Use Case the Application asks the Camera System for information about internal details. The Camera System delivers all information that is needed to describe itself on a static level. This information includes

- available cameras (type and features)
- available storage media (type and features)

and should be obtained once at the beginning of the object lifecycle.

2.2.1.1 detailed description of camera information

The camera information consists of following data

1	**availability**	Availability at this level means that the camera is physically attached to the system and the system is configured to be able to use the camera. Any kind of error state or malfunctioning of the camera does not influence the static availability of the camera.	
2	**type**	**room-camera**	focus on complete self-service-area
		person-camera	focus on person standing in front of the terminal
		exit-slot-camera	focus on exit slot(s) of the terminal
		other	focus is vendor specific

The type of the camera is defined on a logical level.
It can be implemented by different cameras or by one single camera with different focus.

2.2.1.2 detailed description of storage media information

This information contains following data:

1	**availability of storage media**	The storage media can simply be available or not. Availability at this level means that the media is physically attached to the system and the system is configured to be able to use it. Any kind of error state or malfunctioning of the media does not influence the static availability of the media	
2	**type of add-data-feature**	**none**	no data can be added
		automatic	data is automatically added to the picture
		manual	data can be manually added
3	**capacity of storage media**	maximum number of pictures that can be stored on the media	
4	**maximum length of data**	specifies the maximum length of the data that is displayed on the photo. Zero, if data cannot be manually added to the picture	

2.2.2 Use Case 2: Get Status Information

In this Use Case the Application asks the Camera System for information about the current state of the system. The Camera System delivers all information that is needed to describe itself on a dynamic level. This information includes

- current state of available cameras
- current state of available storage media

and can change during the lifecycle of the objects.
It should be obtained whenever needed for any action.

2.2.2.1 detailed description of camera status

The camera status is completely covered by the JxfsStatus class.
The table shows the different states of a single camera and the mapping to corresponding methods of the JxfsStatus-class:

	state	description	JxfsStatus
1	**Online**	the device is present and operational (i.e. not busy processing a request and does not have a hardware error).	***isWorking***
2	**Offline**	The device is present and powered on but is not operational (e.g. a switch may have been used to change it into offline-state)	***!isWorking***
3	**Powered Off**	The device is present but is currently powered off	***isPowerSave***
4	**Busy**	The device is present and a request is currently being processed	***isBusy***
5	**No Device**	There is no device connected	***(not needed)***
6	**Hardware Error**	The device is present but a hardware fault prevents it from being used	***isHardwareError***
7	**User Error**	The device is present but a person is preventing proper operation. The application should suspend the device operation or remove the device from service until the service provider generates a device change event indicating the condition of the device has changed i.e. the error is removed.	***isUserActionError***

The status of each single camera must be available separately.

2.2.2.2 detailed description of storage media status

The status of the storage media is completely covered by the JxfsMediaStatus and the JxfsThresholdStatus classes.

2.2.3 Use Case 3: Take Picture

In this Use Case the Application instructs the Camera System to execute the "take picture" command. The camera system performs this task and replies accordingly.
There are two variants of this task:

1	**take picture without data**	only takes picture without inserting data manually also applicable for cameras with automatic data insert
2	**take picture with data**	takes picture and inserts data manually

The data to be inserted is defined as a String.
The definition of the data format is vendor specific.

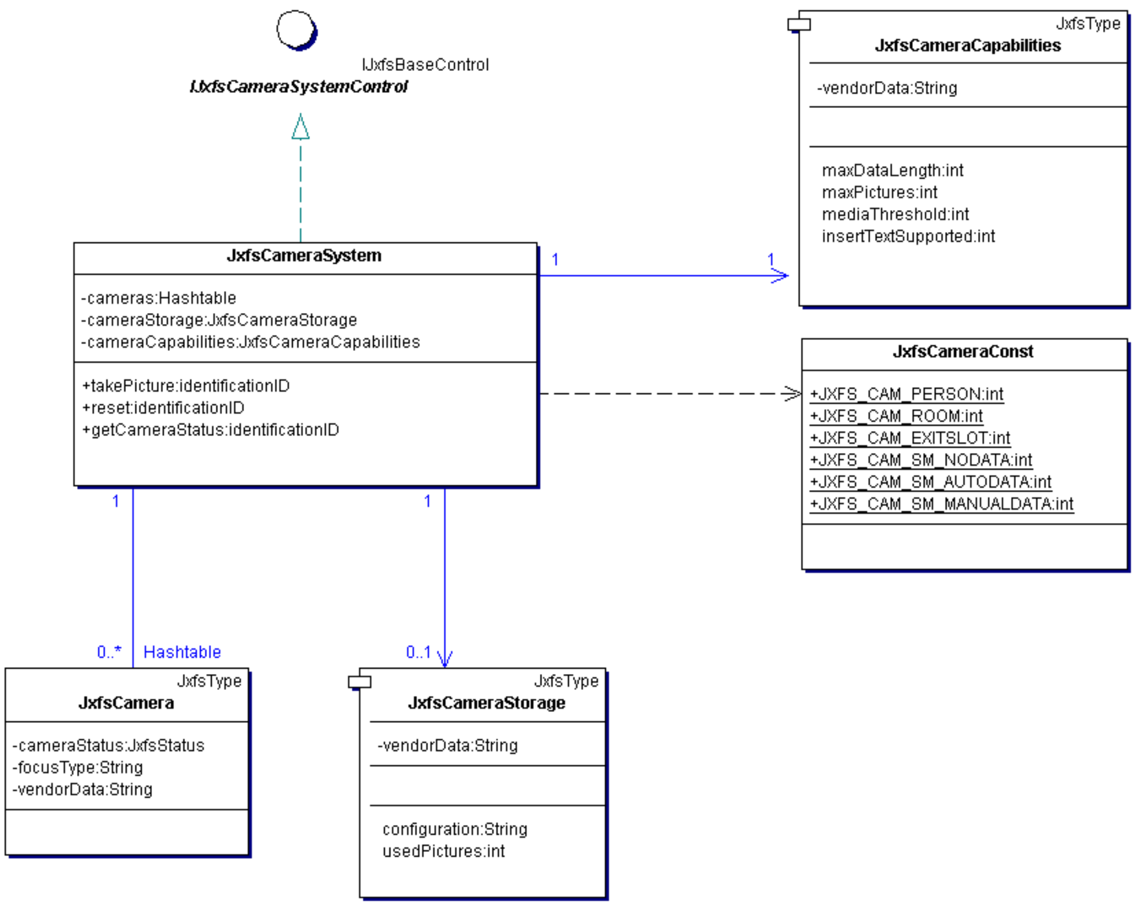
2.3 Classes and Interfaces

As stated previously, the Camera Device Control class allows access to CAM type devices. An overview of the device operation is described in this section from the point of view of the application or applet (referred to as just an application).

2.3.1 Description of the main architecture of the camera system interface

An application obtains an instance of JxfsCameraSystem and then uses the available methods to do I/O. If an error occurs in initiating the I/O, a JxfsException will be thrown. The application should be designed to catch and handle the errors thrown. When no error occurs then control will be returned to the application and an JxfsEvent will be used to signal I/O completion asynchronous to the invoking applications thread of execution.
As a result of the event based I/O operation model, an application will have to register itself as a listener with the JxfsCameraSystemControl object for the event(s) generated.

The following diagram shows the interface from the application point of view:



Class or Interface	Name	Description	Extends / Implements
Interface	**IJxfsCameraSystemControl**	Base interface for all camera systems	**IJxfsBaseControl**
Class	**JxfsCameraSystem**	Base class for all Camera controls.	Extends: **IJxfsCameraSystemControl** **JxfsBaseControl**
Class	**JxfsCamera**	Base class for all Camera devices.	extends: **JxfsType**
Class	**JxfsCameraStorage**	Base class for all Camera Storages	extends: **JxfsType**

The Camera Device Control class is defined in the JxfsCameraSystemControl class and is derived from the class JxfsBaseControl. It contains the methods and properties specific to all the device controls for the Camera device category.

2.4 Support Classes

Class or Interface	Name	Description	Extends / Implements
Interface	**IJxfsConst**	Interface containing the J/XFS constants that are common to several device categories	—
Interface	**IJxfsCameraSystemConst**	Interface containing the J/XFS constants that are common to all the Camera device controls.	—
Class	**JxfsCameraCapabilities**	contains all infos about capabilities of the camera system	extends: **JxfsType**

2.5 Handling of *null* parameters

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

3 Device behavior

3.1 Device open()

During the device open call the Device Service tries to access the connected device. This may fail in the following circumstance:

JXFS_E_HARDWAREERROR	If the device could not be accessed. This may be that the device is not connected or broken. This error should only be issued, if the device service does not see a reasonable chance to make the device work again. For (maybe temporary) error conditions, the open should succeed but the device status should indicate the error condition.
JXFS_E_OPEN	The open was already done by this Device Control

3.2 Status event

If a device status changes a StatusEvent is sent.
This might happen under the following conditions:

- the state of the complete camera system has changed
- the state of the storage media has changed
- the state of the storage media threshold has changed
- the state of a single camera has changed

StatusEvent.getStatus() returns:	**StatusEvent.getDetails() returns:**
JXFS_S_CAM_STATUS_CHANGED	JxfsStatus
JXFS_S_CAM_MEDIA_CHANGED	JxfsMediaStatus
JXFS_S_CAM_THRESHOLD_CHANGED	JxfsThresholdStatus
JXFS_S_CAM_CAMERA_CHANGED	JxfsCamera

The d*etails*-property of the StatusEvent returns different classes that can be used to query further details using the specific methods of these classes.

4 Classes and Interfaces Details

4.1 IJxfsCameraSystemControl

4.1.1 Introduction

The J/XFS Camera Device Control class is defined in IJxfsCameraSystemControl. The intent of the J/XFS Camera Device Control is to allow data and control to pass between the application and the device support code so that the associated device can be accessed.

4.1.2 Summary

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 send the J/XFS Camera Device Control object the appropriate JavaBeans conform method. In case of a property of type boolean an isProperty() method is used to read the property.

Extends: IJxfsBaseControl

Properties

Property	Type	Access	Initialized by
cameras	Hashtable of JxfsCamera	R	Device service
cameraCapabilities	JxfsCameraCapabilities	RW	Device service
cameraStorage	JxfsCameraStorage	R	Device service

Methods

Method	Return	Meaning
takePicture	identificationID	takes picture with specified camera with or without additional text
reset	identificationID	resets cameras or camera system
getCameraStatus	identificationID	returns status of cameras

The common **exceptions** thrown by all methods are:

Value	Meaning
JXFS_E_CLOSED	The Device Control has not been opened.
JXFS_E_UNREGISTERED	The device is not registered at the JxfsDeviceManager
JXFS_E_REMOTE	A network error occurred
JXFS_E_PARAMETER_INVALID	Parameter passed to method is invalid.
JXFS_E_NOT_SUPPORTED	Method is not supported.

4.1.3 Properties

4.1.3.1 cameras (R)

Type	**Hashtable**
Remarks	provides access to the single cameras elements**:** type JxfsCamera keys: focusType (String) if focusType is a standard focus type use predefined constants

4.1.3.2 cameraCapabilities (RW)

Type	**JxfsCameraCapabilities**
Remarks	used to keep complete information about camera capabilities

4.1.3.3 cameraStorage (RW)

Type	**JxfsCameraStorage**
Remarks	offers complete access to the storage media of the camera system

4.1.4 Methods

4.1.4.1 takePicture()

Syntax ***identificationID takePicture(String focusType) throws JxfsException; identificationID takePicture(String focusType, String cameraText) throws JxfsException;***

Description This method will initiate the camera system to take a photograph. Furthermore data can be sent to be displayed on the photo.
The method returns an identificationID that identifies this operation.

focusType specifies the individual camera, which should be used to take the picture. Several standard focus types are defined in IJxfsCameraConst (See subsection IJxfsCameraConst Interface).

cameraText specifies the text string to be displayed on the photo. If the maximum text length is exceeded then the data will be truncated. In this case an event is generated to notify the error. Netherless the picture is taken.

Events

JxfsOperationCompleteEvent This method requires I/O. Upon successful completion it will result in an JxfsOperationCompleteEvent having a status value of:

Field	**Value & Meaning**
OperationID	JXFS_O_CAM_TAKEPICTURE
IdentificationId	The corresponding Id for the completed operation.
Result	JXFS_RC_SUCCESSFUL The operation was completed successfully. JXFS_RC_ UNSUCCESSFUL The operation was not completed with success. JXFS_E_CAM_NOT_SUPPORTED The specified camera is not supported . JXFS_E_CAM_ DATA_TRUNCATED The maximum text length was exceeded, the text was truncated. Nevertheless the picture was taken JXFS_E_CAM_ MEDIATHRESHOLD The state of the recording media reached a threshold after the picture was taken. JXFS_E_CAM_ MEDIAFULL The recording media is full after the picture was taken.
Data	*JxfsType* object equals *null*

Exceptions No additional exceptions thrown.

4.1.4.2 reset()

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

Description This method is used to reset the camera system and put.it into a defined operational state.

Events

JxfsOperationCompleteEvent This method requires I/O. Upon successful completion it will result in an JxfsOperationCompleteEvent having a status value of:

Field	**Value & Meaning**
OperationID	JXFS_O_CAM_RESET
IdentificationId	The corresponding Id for the completed operation.
Result	JXFS_RC_SUCCESSFUL The operation was completed successfully JXFS_RC_ UNSUCCESSFUL The operation was not completed successfully
Data	*JxfsType* object equals *null*

Exceptions No additional exceptions thrown.

4.1.4.3 getCameraStatus()

Syntax ***identificationID getCameraStatus(String focusType) throws JxfsException;***

Description This method returns the status of the specified camera.

focusType specifies the individual camera.

Events

JxfsOperationCompleteEvent This method requires I/O. Upon successful completion it will result in an JxfsOperationCompleteEvent having a status value of:

Field	**Value & Meaning**
OperationID	JXFS_O_CAM_GETCAMSTATUS
IdentificationId	The corresponding Id for the completed operation.
Result	JXFS_RC_SUCCESSFUL The operation was completed successfully JXFS_RC_ UNSUCCESSFUL The operation was not completed successfully
Data	*JxfsStatus* object contains status of camera

Exceptions No additional exceptions thrown.

4.2 JxfsCamera

4.2.1 Introduction

This class identifies the a single camera device.

4.2.2 Summary

Extends: JxfsType

Property	Type	Access	Initialized by
CameraStatus	JxfsStatus	RW	device service
FocusType	String	R	device service
VendorString	String	R	device service

4.2.3 Properties

4.2.3.1 cameraStatus (RW)

Type	**JxfsStatus**
Remarks	standard JxfsStatus type, no additional status required

4.2.3.2 focusType (R)

Type	**String**
Remarks	indicates the place where the camera is focussed to. Several standard focus types are defined in IJxfsCameraConst (See subsection IJxfsCameraConst Interface). For other nonstandard focusses short and descriptive Strings should be used. This String is also used as the key to access the elements of the camera Hashtable. The used focus is fix and cannot be changed during runtime, only by configuration.

4.2.3.3 vendorString (R)

Type	**String**
Remarks	any kind of vendor specific information

4.3 JxfsCameraStorage

4.3.1 Introduction

This class identifies the storage media device that is attached to the camera-system.

4.3.2 Summary

Extends: JxfsType

Properties

Property	Type	Access	Initialized by
VendorData	String	R	device service
UsedPictures	int	R	device service
configuration	String	RW	device service

4.3.3 Properties

4.3.3.1 vendorData (R)

Type	**String**
Remarks	any kind of vendor specific information

4.3.3.2 usedPictures (R)

Type	**String**
Remarks	number of pictures already stored on the storage media

4.3.3.3 configuration (R)

Type	**String**
Remarks	any kind of vendor specific configuration data

4.4 JxfsCameraCapabilities

4.4.1 Introduction

This class identifies the storage media device that is attached to the camera-system.

4.4.2 Summary

Extends: JxfsType

Properties

Property	Type	Access	Initialized by
maxDataLength	int	R	device service
MaxPictures	int	R	device service
mediaThreshold	int	RW	device service
insertTextSupported	int	R	device service

4.4.3 Properties

4.4.3.1 maxDataLength (R)

Type	**String**
Remarks	maximal length of string to be added to the picture

4.4.3.2 maxPictures (R)

Type	**String**
Remarks	maximal number of pictures that can be stored on the storage media

4.4.3.3 mediaThreshold (RW)

Type	**String**
Remarks	maximal number of pictures that can be stored on the storage media

4.4.3.4 insertTextSupported (R)

Type	**String**
Remarks	indicates the type of support for inserting text to the picture. Possible values are defined in *IJxfsCameraConst* (See subsection IJxfsCameraConst Interface).

5 General Classes and Interfaces

5.1 IJxfsCameraConst Interface

5.1.1 Introduction

This interface defines all CAM specific constants. For common constants please refer to the J/XFS Base Architecture.

5.1.2 Constants

standard focus types:

Value	Meaning
JXFS_CAM_PERSON	focus of camera is on person
JXFS_CAM_ROOM	focus of camera is on room
JXFS_CAM_EXITSLOT	focus of camera is on exit slot

types of text support of storage media:

Value	Meaning
JXFS_CAM_SM_NODATA	insert text not supported
JXFS_CAM_SM_AUTODATA	text is inserted automatically
JXFS_CAM_SM_MANUALDATA	text can be inserted manually

Device specific operationID sent with events:

Value	Meaning
JXFS_O_CAM_RESET	Indicates the *reset* operation was completed successfully.
JXFS_O_CAM_TAKEPICTURE	Indicates the *takePicture* operation was completed successfully.
JXFS_O_CAM_GETCAMSTATUS	Indicates the *getCameraStatus* operation was completed successfully.

Status Event codes:

Value	Meaning
JXFS_S_CAM_STATUS_CHANGED	The status has changed.
JXFS_S_CAM_CAMERA_CHANGED	The status of a camera has changed.
JXFS_S_CAM_MEDIA_CHANGED	The status of the storage media has changed.
JXFS_S_CAM_THRESHOLD_CHANGED	The status of media threshold has changed.

Device specific error codes:

Value	Meaning
JXFS_RC_SUCCESSFUL	The operation was completed successfully
JXFS_RC_UNSUCCESSFUL	The operation was not completed successfully
JXFS_E_CAM_NOT_SUPPORTED	The specified camera is not supported
JXFS_E_CAM_DATA_TRUNCATED	The maximum text length was exceeded, the text was truncated. Nevertheless the picture was taken
JXFS_E_CAM_MEDIATHRESHOLD	The state of the recording media reached a threshold after the picture was taken
JXFS_E_CAM_MEDIAFULL	The recording media is full after the picture was taken
JXFS_E_CAM_TAKEPICTURE	Indicates the *takePicture* operation completed with an error.
JXFS_E_CAM_RESET	Indicates the *reset* operation completed with an error.

5.2 Numerical values

Value	Meaning
“Person”	JXFS_CAM_PERSON
“Room”	JXFS_CAM_ROOM
“ExitSlot”	JXFS_CAM_EXITSLOT
“NoData”	JXFS_CAM_SM_NODATA
“AutoData”	JXFS_CAM_SM_AUTODATA
“ManualData”	JXFS_CAM_SM_MANUALDATA
14000	JXFS_O_CAM_RESET
14001	JXFS_O_CAM_TAKEPICTURE
14002	JXFS_O_CAM_GETCAMSTATUS
14003	JXFS_S_CAM_STATUS_CHANGED
14004	JXFS_S_CAM_CAMERA_CHANGED
14005	JXFS_S_CAM_MEDIA_CHANGED
14006	JXFS_E_CAM_NOT_SUPPORTED
14007	JXFS_E_CAM_DATA_TRUNCATED
14008	JXFS_E_CAM_MEDIATHRESHOLD
14009	JXFS_E_CAM_MEDIAFULL
14010	JXFS_E_CAM_TAKEPICTURE
14011	JXFS_E_CAM_RESET

6 APPENDIX A : CEN/ISSS WORKSHOP 14923:2004 CORE MEMBERS :

DELARUE

DIEBOLD

DYNASTY

IBM

KAL

KEBA

LUTZ WOLF GRUPPE

NCR

NEXUS

SEIKO EPSON CORPORATION

WINCOR - NIXDORF