CEN

WORKSHOP

AGREEMENT

CWA 14923-4

May 2004

ICS 35.240.40

Supersedes CWA 13937-4:2003

English version

J/eXtensions for Finalcial Sevices (J/XFS) for the Java Platform - Part 4: Text Input/Output Device Interface - Programmer's Reference

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

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

Ref. No.:CWA 14923-4:2004 E

Contents

CONTENTS 2

FOREWORD 3

HISTORY 4

1. SCOPE 5

2. OVERVIEW 6

2.1. DESCRIPTION 6
2.2. CLASS HIERARCHY 7
2.3. CLASSES AND INTERFACES 8
2.4. SUPPORT CLASSES 8

3. DEVICE BEHAVIOR 9

3.1. DEVICE OPEN() 9
3.2. HANDLING OF NULL PARAMETERS 9

4. CLASSES AND INTERFACES 10

4.1. TEXT INPUT/OUTPUT INTERFACE IJXFSTIOCONTROL 10
4.1.1. Introduction *10*
4.1.2. Summary *10*
4.1.3. Properties *11*
4.1.4. Methods *13*
4.2. TEXT INPUT / OUTPUT INTERFACE IJXFSTIOSERVICE 19
4.3. TEXT INPUT/OUTPUT CLASS JXFSTIO 19
4.4. JXFSTIOSTATUS CLASS 19
4.4.1. Introduction *19*
4.4.2. Summary *19*
4.4.3. Properties *20*
4.5. JXFSTIORESOLUTION CLASS 20
4.5.1. Introduction *20*
4.5.2. Summary *20*
4.5.3. Properties *21*
4.6. JXFSTIOCONST INTERFACE 21
4.6.1. Introduction *21*
4.6.2. Constants *21*

5. APPENDIX A : CEN/ISSS WORKSHOP 14923:2004 CORE MEMBERS : 24

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 14923-4:2004 replaces CWA 13937-4:2003 and should be read in conjunction with CWA 13937-4:2000, 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 main differences to the previous CWA 13937:2000 are:

- Modified the description for the readKeyboardData method.
- Changed the description for the readKeyboardData flush parameter.
- Changed the description for the readKeyboardData autoEnd parameter.
- Added paragraph specifying handling of null parameters.
- Added a class hierarchy diagram
- Removed the JXFS_E_CLAIMED exception
- Modified the Description of the IJxfsTIOControl’s resolutionProperty
- Modified the Description of the IJxfsTIOControl’s beep method
- Modified the Description of the IJxfsTIOControl’s getLED method
- Modified the Description of the IJxfsTIOControl’s clearScreen method
- Modified the Description of the IJxfsTIOControl’s writeDisplayData method
- Modified the Description of the IJxfsTIOControl’s readKeyboardData method
- Modified the comment for the parameter “numOfChars” passed into the readKeyboardData call
- Modified the comment for the field “data” of the OperationCompleteEvent of the readKeyboardData
- Modified the access of the JxfsTIOStatus’s properties

1. Scope

This document describes the Text Input / Output Device Class (TIO) 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 Text I/O 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

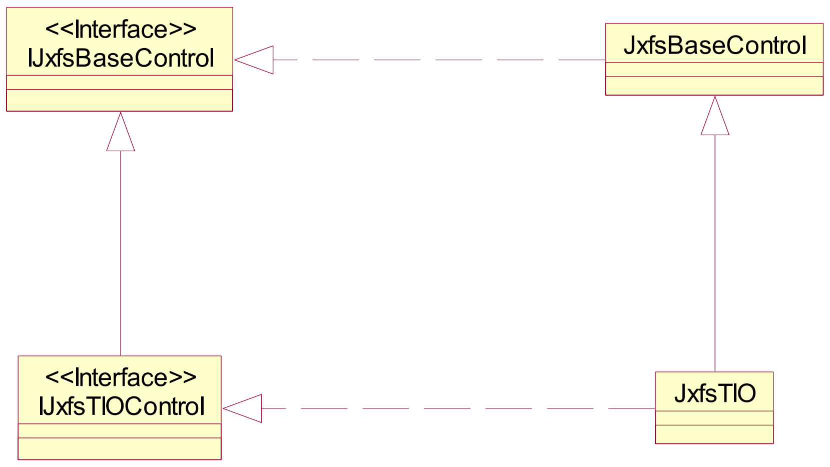
2.1. Description

The Text Input Output Device Control class, defined in the JxfsTIO class, is a subclass of the JxfsBaseControl class. Its interface is defined in the IJxfsTIOControl class which is a subclass of the IJxfsBaseControl class. The intended use of an Text Input Output object is to allow data and control to be passed between a Java application or applet and a TIO type device so that the associated device can be accessed through a "Pure Java" platform.

As stated previously, the Text Input Output Device Control class allows access to TIO 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).
An application will instantiate a JxfsTIO object and then use the available methods to do I/O. If an error occurs in initiating the I/O, an JxfsException will be thrown. The application should be designed to catch and handled the errors thrown. Otherwise, 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 JxfsTIO object for the event(s) generated.

This document describes the input and output features of the TIO. It offers the functionality of a text display, a set of LEDs and a beep mechanism. In addition function keys and a tiny keyboard are also supported.

2.2. Class Hierarchy



2.3. Classes and Interfaces

Class or interface	Name	Description	Extends / Implements
Interface	**IJxfsBaseControl**	Base interface for all device controls. Contains methods specific to all the device controls.	—
Class	**JxfsBaseControl**	Base class for all device controls. Implements methods common for all devices.	—
Interface	**IJxfsTIOControl**	Base interface for all Text Input/Output controls. Contains the methods specific to all the device controls for the Text Input/Output device category.	Extends: **IJxfsBaseControl**
Interface	**IJxfsTIOService**	Base interface for all Text Input/Output services. Contains the methods specific to all the device services for the Text Input/Output device category.	Extends: **IJxfsBaseService**
Class	**JxfsTIO**	Base class for all Text Input/Output controls Implements the methods defined in the **IJxfsTIOControl** Interface. Contains the properties specific to all Text Input/Output device controls.	Extends: **JxfsBaseControl** Implements: **IJxfsTIOControl**

2.4. Support Classes

Class or interface	Name	Description	Extends / Implements
Interface	**JxfsConst**	Interface containing the J/XFS constants that are common to several device categories	—
Interface	**JxfsTIOConst**	Interface containing the J/XFS constants that are common to all the Text Input/Output device controls.	—
Class	**JxfsTIOStatus**	Describes the TIO specific status information.	Extends: **JxfsStatus**
Class	**JxfsTIOResolution**	Keeps the resolution (in characters per row and column).	Extends: **JxfsType**

3. Device behavior

3.1. Device open()

During the device open call the Device Service tries to access the connected device. This fails for the following circumstances:

JXFS_E_HARDWAREERROR	If the device could not be accessed. This may be that the device is not connected or broken.
JXFS_E_OPEN	The open was already done by this Device Control.

3.2. Handling of null parameters

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

4. Classes and Interfaces

4.1. Text Input/Output Interface IJxfsTIOControl

4.1.1. Introduction

The J/XFS Text Input/Output Device Control interface is defined in IJxfsTIOControl and extends of IJxfsBaseControl. The intent of the J/XFS Text Input/Output 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.

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 Text Input/Output Device Control object the appropriate JavaBeans conform method.

It should not be assumed that the device has a clear screen after an open.

Extends: IJxfsTIOControl

Properties

Property	**Type**	**Access**	**Initialized by**
cursor	boolean	R	device service
status	JxfsTIOStatus	R	device service
resolution	JxfsTIOResolution	R/W	device service
availableResolutions	Vector	R	device service
displayLight	boolean	R	device service
beep	boolean	R	device service
maxLED	int	R	device service
keyboard	boolean	R	device service
keyboardLock	boolean	R	device service

The common exceptions thrown by all property 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

Methods

Method	**Return**	**Meaning**
beep	int	Sounds a beep signal.
lightDisplay	int	Lights the text display.
setLED	int	Lights the specified LED.
getLED	int	Gets the current light type of specified LED.
clearScreen	int	Clears display screen.
writeDisplayData	int	Writes data on display.
isTextAttributeSupported	boolean	Detects supported text attributes.
readKeyboardData	int	Reads pressed keys.

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

cursor Property R

Type	*boolean*
Initial Value	-
Description	Specifies whether the Text Input Output device has a display cursor. The value can be true or false depending on the characteristics of the display.
Method	*isCursorSupported()*
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

status Property R

Type	*JxfsTIOStatus*
Initial Value	-
Description	Depending on the state of the Text Input Output device, the status object will be updated. For details on JxfsTIOStatus please see the appropriate section.
Method	*getStatus()*
Events	If the values of these properties kept by the status object changes the device service will send all registered StatusListeners a StatusEvent with *status* = JXFS_S_TIO_STATUS_CHANGED. The status object is attached in the *details* field.
Exceptions	No additional exceptions thrown.

resolution Property R/W

Type	*JxfsTIOResolution*
Initial Value	-
Description	Specifies the horizontal and vertical size of the display in character columns and rows. If no resolution is set or an unsupported resolution is specified the default resolution will be used. After redefining resolution and before displaying a new text the display should be cleared to assure proper text output. If no screen is available a resolution of 0,0 is returned.
Method	*getResolution(), setResolution(JxfsTIOResolution res)*
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

availableResolutions R

Type	*Vector*
Initial Value	-
Description	Specifies available display resolutions. All resolutions are kept in a Vector consisting of JxfsTIOResolution objects.
Method	*getAvailableResolution()*
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

displayLight Property R

Type	*boolean*
Initial Value	**-**
Description	Specifies whether the Text Input Output device supports display light. The value can be true or false depending on the characteristics of the device.
Method	*isDisplayLightSupported ()*
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

beep Property R

Type	*boolean*
Initial Value	**-**
Description	Specifies whether the Text Input Output device supports beeping. The value can be true or false depending on the characteristics of the device.
Method	*isBeepSupported ()*
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

maxLED Property R

Type	*int*
Initial Value	**-**
Description	Specifies the number of LED‘s supported by the Text Input Output device.
Method	*getMaxLED()*
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

keyboard Property R

Type	*boolean*
Initial Value	**-**
Description	Specifies if a keyboard is supported. The value is *true* if available, *false* otherwise.
Method	*isKeyboardSupported ()*
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

keyboardLock Property R

Type	*boolean*
Initial Value	**-**
Description	Specifies if a keyboardLock is supported. The value is *true* if available, *false* otherwise.
Method	*isKeyboardLockSupported ()*
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

4.1.4. Methods

beep() Method

Syntax ***int beep(int beepValue, int time) throws JxfsException;***

Description This method can be used to set the conditions for sounding a beep (in case of _BEEP_KEYPRESS) or for actually sounding a beep. Returns an identificationID that identifies this operation.

beepValue can be one of the following:

Value	**Meaning**
JXFS_TIO_BEEP_OFF	The beeper is turned off.
JXFS_TIO_BEEP_KEYPRESS	The beeper will sound on key press.
JXFS_TIO_BEEP_CONTINUOUS	The beeper sounds continuously.
JXFS_TIO_BEEP_EXCLAMATION	The beeper sounds an exclamation signal.
JXFS_TIO_BEEP_WARNING	The beeper sounds an warning signal.
JXFS_TIO_BEEP_ERROR	The beeper sounds an error signal.
JXFS_TIO_BEEP_CRITICAL	The beeper sounds an critical error signal.

time in milliseconds. If the value is greater than zero the TIO will beep for the specified time. If equal to JXFS_FOREVER, beeping is performed forever. If *beep()* is called a second time the current beeping ends always immediately (e.g. with *beepValue* equal to JXFS_TIO_BEEP_OFF or with a new specified time).
If this method is called with a beepValue of JXFS_TIO_BEEP_OFF the time parameter is ignored and key press beeping will also be switched off (if it was on).
A time of JXFS_FOREVER is valid for the beepValue of JXFS_TIO_BEEP_KEYPRESS.
The operation complete event for this method will return immediately once the beep has been initiated at the device level. The reasons for this approach are:

If you specify an infinite time, an operation complete event will never return if the beep is not cancelled or another beep is not issued.

If you assume that operation compete events are issued after the completion of the beep, then this might also have an impact on the scheduling of commands. If you have a simple scheduling implementation where the next scheduled command will be run after the previous has finished, then you could never stop a beep with an infinite time with another beep.

There are many peripheral devices that themselves take beep commands with a time constraint. For these devices implementations are a lot easier, if you issue the beep command to the device and let the device do the rest; but only very few terminals have the ability to return information whether they are still beeping or not. So it is easier, in many cases, for implementations if the operation complete event is issued earlier than the end of the beep.

Therefore, it is assumed that the operation complete event for this method indicates whether the beep was initiated successfully; and does not reflect whether or not it was cancelled by a succeeding beep method call.

Events

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

Field	**Value & Meaning**
operationID	JXFS_O_TIO_BEEP
identificationId	The corresponding Id for the completed operation.

result	JXFS_RC_SUCCESSFUL The operation was completed with success. JXFS_E_CANCELLED The operation was cancelled. JXFS_E_TIO_BEEP Indicates the operation completed with an error.
data	*JxfsType* object equals *null*

Exceptions No additional exceptions thrown.

lightDisplay() Method

Syntax ***int lightDisplay(boolean on) throws JxfsException;***

Description This method can be used to switch display lighting on (***on*** equals true) or off (***on*** equals false). Returns an identificationID that identifies this operation.

Events

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

Field	**Value & Meaning**
operationID	JXFS_O_TIO_LIGHT
identificationId	The corresponding Id for the completed operation.
result	JXFS_RC_SUCCESSFUL The operation was completed with success. JXFS_E_CANCELLED The operation was cancelled. JXFS_E_TIO_LIGHT Indicates the operation completed with an error.
data	*JxfsType* object equals *null*

Exceptions No additional exceptions thrown.

setLED() Method

Syntax ***int setLED(int index, int type) throws JxfsException;***

Description This method can be used for lighting a LED. It returns an identificationID that identifies this operation.

type can be one of the following:

Value	**Meaning**
JXFS_TIO_LED_OFF	The LED is turned off.
JXFS_TIO_LED_CONTINUOUS	The LED is turned on continuously.
JXFS_TIO_LED_SLOWFLASH	The LED is set to flash slowly.
JXFS_TIO_LED_MEDIUMFLASH	The LED is blinking medium frequency.
JXFS_TIO_LED_QUICKFLASH	The LED is set to flash quickly.

index Specifies which LED to light. If it is equal to a value from 1 to maxLED() the LED with the appropriate index will be lighted. In addition to specifying the number of the LED it can be equal to one of the following values:

Value	**Meaning**
JXFS_TIO_LED_ERROR	The error LED will be lighted.
JXFS_TIO_LED_WARNING	The warning LED will be lighted.
JXFS_TIO_LED_ONLINE	The online LED will be lighted.
JXFS_TIO_LED_OFFLINE	The offline LED will be lighted (or the online LED turns off).
JXFS_TIO_LED_NORMAL	Indicates proper working of the device
JXFS_TIO_LED_PAPERLOW	The paper low LED will be lighted.

JXFS_TIO_LED_PAPEREMPTY	The paper empty LED will be lighted.
JXFS_TIO_LED_PAPERJAM	The paper jam LED will be lighted.
JXFS_TIO_LED_TONERLOW	The toner low LED will be lighted.
JXFS_TIO_LED_TONEREMPTY	The toner empty LED will be lighted.

Events

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

Field	**Value & Meaning**
operationID	JXFS_O_TIO_LED
identificationId	The corresponding Id for the completed operation.
result	JXFS_RC_SUCCESSFUL The operation was completed with success. JXFS_E_CANCELLED The operation was cancelled. JXFS_E_TIO_LED Indicates the operation completed with an error.
data	*JxfsType* object equals *null*

Exceptions No additional exceptions thrown.

getLED() Method

Syntax ***int getLED(int index) throws JxfsException;***

Description This method can be used to query the current lighting of an LED. Returns a type code specifying the lightning status. Throws an exception JXFS_E_PARAMETER_INVALID. The valid values of the index parameter are the same as those specified in the settled method.
The returned integer is one of the following:

Value	**Meaning**
JXFS_TIO_LED_OFF	The LED is turned off.
JXFS_TIO_LED_CONTINUOUS	The LED is turned on continuously.
JXFS_TIO_LED_SLOWFLASH	The LED is set to flash slowly.
JXFS_TIO_LED_MEDIUMFLASH	The LED is blinking medium frequency.
JXFS_TIO_LED_QUICKFLASH	The LED is set to flash quickly.

clearScreen() Method

Syntax ***int clearScreen(int positionX, int positionY, int width, int height) throws JxfsException;***

Description This method can be used to clear the display screen. All parameters are in column positions. Returns an identificationID that identifies this operation. The positionX and positionY parameters indicate the top left corner of the area to be cleared. A one based co-ordinate system where the display origin is at (1,1) in the top left of the display is used, with X and Y co-ordinates increasing to the right and down respectively. The current position of the cursor after the clear screen operation is the same as it was before this operation.

positionX specifies the starting horizontal position of the area to be cleared.

positionY specifies the starting vertical position of the area to be cleared.

width specifies the horizontal width of the area to be cleared.

height specifies the vertical height of the area to be cleared.

Events

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

Field	**Value & Meaning**
operationID	JXFS_O_TIO_CLEAR
identificationId	The corresponding Id for the completed operation.
result	JXFS_RC_SUCCESSFUL The operation was completed with success. JXFS_E_CANCELLED The operation was cancelled. JXFS_E_TIO_CLEAR Indicates the operation completed with an error.
data	*JxfsType* object equals *null*

Exceptions No additional exceptions thrown.

writeDisplayData() Method

Syntax ***int writeDisplayData(int mode, int posX, int posY, int textAttr, String text) throws JxfsException;***

Description This method can be used to write text to the display. The text is wrapped automatically on the end of the line, except on the last one, where text is truncated. Returns an identificationID that identifies this operation.
When relative positioning is selected posY and posX can be 0, meaning write at the current output position. If posY or posX are less than 0 output is written above or to the left of the current output position respectively.
When writing more data then can be displayed on the last display line, the data being output is truncated and the output position for the next write command is set to the origin (top left) of the display.

mode Specifies the mode of text positioning. It can be one of the following:

Value	**Meaning**
JXFS_TIO_POS_RELATIVE	The text is positioned relative to current position.
JXFS_TIO_POS_ABSOLUTE	The text is positioned to an absolute position.

posX Specifies the starting horizontal position to display the text. This will be an offset from the current position for relative mode and a position value for absolute mode, where value 1 means the most left position.

posY Specifies the starting vertical position to display the text. This will be an offset from the current position for relative mode and a position value for absolute mode, where value 1 means the most top position.

textAttr Specifies the text attributes of the text being displayed. It can have a combination of the following values:

Value	**Meaning**
JXFS_TIO_TEXT_NORMAL	The normal text display.
JXFS_TIO_TEXT_UNDERLINED	The text is underlined.
JXFS_TIO_TEXT_INVERTED	The text is displayed light on black.
JXFS_TIO_TEXT_FLASH	The text is displayed flashing.

If one of the modi mentioned above is not supported, another best matching mode is selected. The values can also be combined (i.e. underline and inverted). This is achieved by OR'ing the corresponding values.

text Specifies the text to be displayed.

Events

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

Field	**Value & Meaning**
operationID	JXFS_O_TIO_DISPLAY
identificationId	The corresponding Id for the completed operation.
result	JXFS_RC_SUCCESSFUL The operation was completed with success. JXFS_E_CANCELLED The operation was cancelled. JXFS_E_TIO_DISPLAY Indicates the operation completed with an error.
data	*JxfsType* object equals *null*

Exceptions No additional exceptions thrown.

readKeyboardData() Method

Syntax ***int readKeyboardData(int numOfChars, int mode, int posX, int posY, int echoMode, int echoAttr, int keys, boolean cursor, boolean flush, boolean autoEnd) throws JxfsException***

Description This method can be used to read unformatted text from the keyboard. When complete a Vector containing the keys pressed will be placed in the *data* field of an OperationCompleteEvent and all OperationCompleteListeners will be notified. Returns an identificationID that identifies this operation.
If input is terminated by a function or cancel key, the terminating key is appended to the output data to allow analysis of the termination reason. Therefore, the description of valid keys contained in the data member of the operation complete events returned for this method includes the JXFS_TIO_KEY_CANCEL value. The posX and posY parameters indicate the top left corner of the output position. When absolute positioning is used a one based co-ordinate system, where the display origin is at (1,1) in the top left of the display is used, with X and Y co-ordinates increasing to the right and down respectively. When relative positioning is selected posY and posX can be 0, meaning write at the current output position. If posY or posX are less than 0 output is written above or to the left of the current output position respectively. If echoMode is JXFS_TIO_ECHO_INVISIBLE then the output position is not adjusted as no text is echoed to the display.
When writing to the last display line data being output is truncated and the output position is set to the origin (top left) of the display.

numOfChars Specifies the number of characters to be read from the keyboard.

This parameter does not include any depressions of the delete or backspace keys.

mode Specifies the mode of text positioning. It can be one of the following:

Value	**Meaning**
JXFS_TIO_POS_RELATIVE	The text is positioned relative to current position.
JXFS_TIO_POS_ABSOLUTE	The text is positioned to an absolute position.

posX Specifies the starting horizontal position to display the text. This will be an offset from the current position for relative mode and a position value for absolute mode.

posY Specifies the starting vertical position to display the text. This will be an offset from the current position for relative mode and a position value for absolute mode.

echoMode Specifies the text attributes of the input being echoed. It can have one of the following values:

Value	**Meaning**
JXFS_TIO_ECHO_TEXT	The input will be echoed.

JXFS_TIO_ECHO_INVISIBLE	The input will not be echoed.
JXFS_TIO_ECHO_PASSWORD	The input will echo a replacement character.

echoAttr Specifies the text attributes of the text being echoed. It can have a combination of the following values:

Value	**Meaning**
JXFS_TIO_TEXT_NORMAL	The normal text display.
JXFS_TIO_TEXT_UNDERLINED	The text is underlined.
JXFS_TIO_TEXT_INVERTED	The text is displayed light on black.
JXFS_TIO_TEXT_FLASH	The text is displayed flashing.

If one of the modi mentioned above is not supported, another best matching mode is selected. The values can also be combined (i.e. underline and inverted). This is achieved by OR'ing the corresponding values.

keys Specifies what types of keys the keyboard of the Text Input Output device will allow for input. It may have a value of a combination of the following:

Value	**Meaning**
JXFS_TIO_KEY_NUMERIC	The TIO has numeric keys.
JXFS_TIO_KEY_HEXADECIMAL	The TIO has hexadecimal keys.
JXFS_TIO_KEY_ALPHANUMERIC	The TIO has alphanumeric keys.
JXFS_TIO_KEY_FUNCTION	The TIO has function keys.

The values can also be combined. This is achieved by OR'ing the corresponding values.

cursor Specifies whether the Text Input Output device will display a cursor. The value can be true or false depending on the characteristics of the display.

flush Specifies whether the Text Input Output device will be cleared before input is allowed.

autoEnd Specifies whether input is automatically ended by Device Services when the value given in numOfChars is met. If this is false the input is only ended by pressing the Enter key. The return code is the always successful, even if the numOfChars value specified is not correct.

Events

OperationCompleteEvent When a *readKeyboardData()* operation is completed a OperationCompleteEvent will be sent by the J/XFS TIO Device Control to all registered OperationCompleteListeners. The OperationCompleteEvent will contain the following:

Field	**Value & Meaning**
operationID	JXFS_O_TIO_READ
identificationId	The corresponding Id for the completed operation.
result	JXFS_RC_SUCCESSFUL The operation was completed with success. Only now the data field is filled with the keys pressed. JXFS_E_CANCELLED The operation was cancelled. JXFS_ E_TIO_ READ Indicates the operation completed with an error.
data	A Vector of Integer objects containing the keys read.. This does not contain the final Enter or any Delete keys pressed in between.

The following keys are supported:

Value	Meaning
JXFS_TIO_KEY_0 ... 9	The numeric keys.
JXFS_TIO_KEY_A ... F	The hexadecimal keys.
JXFS_TIO_KEY_DOT	The (.) sign.
JXFS_TIO_KEY_COMMA	The (,) sign.
JXFS_TIO_KEY_SEMICOLON	The (;) sign.
JXFS_TIO_KEY_FENCE	The (#) sign.
JXFS_TIO_KEY_MULTI	The (*) sign.
JXFS_TIO_KEY_SLASH	The (/) sign.
JXFS_TIO_KEY_PLUS	The (+) sign.
JXFS_TIO_KEY_MINUS	The (-) sign.
JXFS_TIO_KEY_F1 ... F10	The function keys.

Exceptions No additional exceptions thrown.

isTextAttributeSupported() Method

Syntax ***boolean isTextAttributeSupported(int textAttr) throws JxfsException;***

Description This method is used to detect supported text attributes.

textAttr Specifies the text attribute the method is detecting. It can have a combination of the following values:

Value	Meaning
JXFS_TIO_TEXT_UNDERLINED	The text is underlined.
JXFS_TIO_TEXT_INVERTED	The text is displayed light on black.
JXFS_TIO_TEXT_FLASH	The text is displayed flashing.

Events No additional events generated.

Exceptions No additional exceptions thrown.

4.2. Text Input / Output Interface IJxfsTIOService

The Device Service interface is common for 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.

4.3. Text Input/Output Class JxfsTIO

This class is the implementation of the interface IJxfsTIOInterface.

4.4. JxfsTIOStatus Class

4.4.1. Introduction

All TIO specific status informations are kept in the JxfsTIOStatus object, that can be queried by using the *getStatus()* method of the JxfsTIO class.

4.4.2. Summary

Extends: JxfsStatus

Property	Type	Access	Initialized by
JxfsTIOStatus()	constructor		-
set*Property(boolean value)*	void		sets the corresponding property
online	boolean	RW	device service
devicePresent	boolean	RW	device service
keyboardOn	boolean	RW	device service
keyboardLockOn	boolean	RW	device service

The constructor initializes all members to false.

4.4.3. Properties

online Property RW

Type	*boolean*
Initial Value	**-**
Description	Returns true if the device is online, false if not.
Method	*isOnline(), setOnline(boolean value)*
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

devicePresent Property RW

Type	*boolean*
Initial Value	-
Description	Returns true if the device is attached to workstation and the power is on, false if not.
Method	*isDevicePresent(), setDevicePresent(boolean value)*
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

keyboardOn Property RW

Type	*boolean*
Initial Value	-
Description	Returns true if the keyboard is activated, false if not.
Method	*isKeyboardOn(), setKeyboardOn(boolean value)*
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

keyboardLockOn Property RW

Type	*boolean*
Initial Value	-
Description	Returns true if the keyboard lock is activated, false if not.
Method	*isKeyboardLockOn(), setKeyboardLockOn(boolean value)*
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

4.5. JxfsTIOResolution Class

4.5.1. Introduction

This class keeps the resolution of the text display. The resolution is described as the number of characters that can be diplayed per row and column.

4.5.2. Summary

Extends: JxfsType

Property	Type	Access	Initialized by
JxfsTIOResolution(int columns, int rows)	constructor		-
set*Property(int value)*	void		sets the corresponding property
columns	int	RW	device service
rows	int	RW	device service

4.5.3. Properties

columns Property RW

Type	*int*
Initial Value	**-**
Description	Returns the number of character per column.
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

rows Property RW

Type	*int*
Initial Value	-
Description	Returns the number of characters per row.
Events	No additional events generated.
Exceptions	No additional exceptions thrown.

4.6. JxfsTIOConst Interface

4.6.1. Introduction

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

4.6.2. Constants

Device specific operationID sent with events:

Value	Meaning
JXFS_O_TIO_BEEP	Indicates the *beep* operation completed with an error.
JXFS_O_TIO_LIGHT	Indicates the *lightDisplay* operation completed with an error.
JXFS_O_TIO_LED	Indicates the *setLED* operation completed with an error.
JXFS_O_TIO_DISPLAY	Indicates the *writeDisplayData* operation completed with an error.
JXFS_O_TIO_READ	Indicates the *readKeyboardData* operation completed with an error.
JXFS_O_TIO_CLEAR	Indicates the *clearScreen* operation completed with an error.

Status Event codes:

Value	Meaning
JXFS_S_TIO_STATUS_CHANGED	The status has changed.

Device specific error codes:

Value	Meaning
JXFS_E_TIO_BEEP	Indicates the *beep* operation completed

	with an error.
JXFS_E_TIO_LIGHT	Indicates the *lightDisplay* operation completed with an error.
JXFS_E_TIO_LED	Indicates the *setLED* operation completed with an error.
JXFS_E_TIO_DISPLAY	Indicates the *writeDisplayData* operation completed with an error.
JXFS_E_TIO_READ	Indicates the *readKeyboardData* operation completed with an error.
JXFS_E_TIO_CLEAR	Indicates the *clearScreen* operation completed with an error.

Method specific constants:

Value	Meaning
JXFS_TIO_BEEP_OFF	The beeper is turned off.
JXFS_TIO_BEEP_KEYPRESS	The beeper will sound on key press.
JXFS_TIO_BEEP_CONTINUOUS	The beeper sounds continuously.
JXFS_TIO_BEEP_EXCLAMATION	The beeper sounds an exclamation signal.
JXFS_TIO_BEEP_WARNING	The beeper sounds an warning signal.
JXFS_TIO_BEEP_ERROR	The beeper sounds an error signal.
JXFS_TIO_BEEP_CRITICAL	The beeper sounds an critical error signal.

Value	Meaning
JXFS_TIO_LED_OFF	The LED is turned off.
JXFS_TIO_LED_CONTINUOUS	The LED is turned on continuously.
JXFS_TIO_LED_SLOWFLASH	The LED is set to flash slowly.
JXFS_TIO_LED_MEDIUMFLASH	The LED is blinking medium frequency.
JXFS_TIO_LED_QUICKFLASH	The LED is set to flash quickly.
JXFS_TIO_LED_ERROR	The error LED will be lighted.
JXFS_TIO_LED_WARNING	The warning LED will be lighted.
JXFS_TIO_LED_ONLINE	The online LED will be lighted.
JXFS_TIO_LED_OFFLINE	The offline LED will be lighted (or the online LED turns off).
JXFS_TIO_LED_NORMAL	Indicates proper working of the device
JXFS_TIO_LED_PAPERLOW	The paper low LED will be lighted.
JXFS_TIO_LED_PAPEREMPTY	The paper empty LED will be lighted.
JXFS_TIO_LED_PAPERJAM	The paper jam LED will be lighted.
JXFS_TIO_LED_TONERLOW	The toner low LED will be lighted.
JXFS_TIO_LED_TONEREMPTY	The toner empty LED will be lighted.

Value	Meaning
JXFS_TIO_POS_RELATIVE	The text is positioned relative to current position.
JXFS_TIO_POS_ABSOLUTE	The text is positioned to an absolute position.

Value	Meaning
JXFS_TIO_TEXT_NORMAL	Normal text.
JXFS_TIO_TEXT_UNDERLINED	The text is underlined.
JXFS_TIO_TEXT_INVERTED	The text is displayed light on black.
JXFS_TIO_TEXT_FLASH	The text is displayed flashing.

The above values are combinable (bitwise OR-able).

Value	Meaning
JXFS_TIO_ECHO_TEXT	The input will be echoed.

JXFS_TIO_ECHO_INVISIBLE	The input will not be echoed.
JXFS_TIO_ECHO_PASSWORD	The input will echo a replacement character.

Keyboard data capabilities

Value	Meaning
JXFS_TIO_KEY_NUMERIC	The TIO has numeric keys.
JXFS_TIO_KEY_HEXADECIMAL	The TIO has hexadecimal keys.
JXFS_TIO_KEY_ALPHANUMERIC	The TIO has alphanumeric keys.
JXFS_TIO_KEY_FUNCTION	The TIO has function keys.

The above values are combinable (bitwise OR-able).

Keyboard data output key definitions

Value	Meaning
JXFS_TIO_KEY_0 ... 9	The numeric keys.
JXFS_TIO_KEY_A ... F	The hexadecimal keys.
JXFS_TIO_KEY_DOT JXFS_TIO_KEY_COMMA JXFS_TIO_KEY_SEMICOLON JXFS_TIO_KEY_FENCE JXFS_TIO_KEY_MULTI JXFS_TIO_KEY_SLASH JXFS_TIO_KEY_PLUS JXFS_TIO_KEY_MINUS JXFS_TIO_KEY_DELETE JXFS_TIO_KEY_CANCEL JXFS_TIO_KEY_ENTER	The (.) sign. The (,) sign. The (;) sign. The (#) sign. The (*) sign. The (/) sign. The (+) sign. The (-) sign. The delete key. The cancel key. The enter key.
JXFS_TIO_KEY_F1 ... F10	The function keys.

5. 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