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

# WORKSHOP AGREEMENT

**CWA 13937-7**

August 2000

ICS 35.240.40

J/eXtensions for Financial Services (J/XFS) for the Java Platform - Part 7: Alarm Device Interface - Programmer's Reference

**Ref. No CWA 13937-7:2000 E**

Rue de Stassart, 36 • B-1050 Bruxelles
Tel : +32 2 550 08 11 • Fax : +32 2 550 08 19

## Foreword

This CWA contains the specifications that define the J/eXtensions for Financial Services (J/XFS) for the Java TM 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 1999-12-15/16 in Geneva and a subsequent electronic review by the Workshop participants, and the final version was sent to CEN for publication on 2000/06-21.

The specification is continuously reviewed and commented in the CEN/ISSS J/XFS Workshop. It is therefore expected that an update of the specification will be published in due time as a CWA, superseding this one. 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/isss/workshop/j-XFS/cwa-updates).

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 CEN/ISSS Secretariat (isss@cenorm.be) who will be forwarding them to the J/XFS Workshop.

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

## Contents

# 1 Scope

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

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

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

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

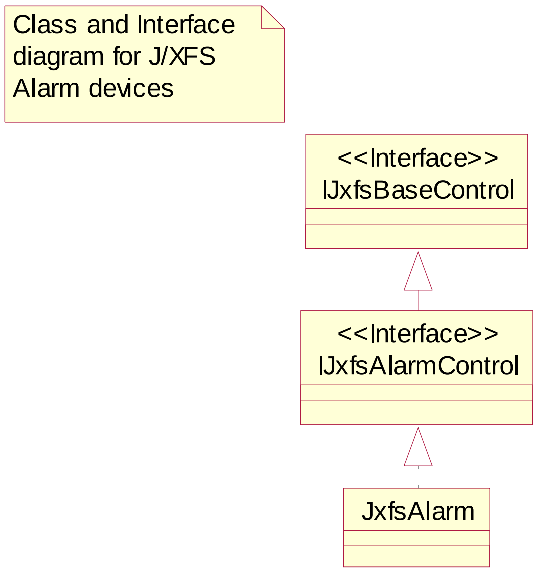
To support alarm devices the basic Device Control structure is extended with various properties and methods specific to this device which are described on the following pages.

## 2 Overview

The Alarm device is used to notify users, devices and other interested parties of security violations. The notification mechanism used by alarm devices to signal such events are device and manufacturer depended.

# 3 Class Diagram

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



# 4 Class and Interface Summary

The following classes and interfaces are used by the J/XFS Alarm Device Control.

| Class or Interface | Name | Description | Extends or Implements |
|---|---|---|---|
| Interface | IJxfsBaseControl | Base interface for all controls. | |
| Interface | IJxfsAlarmControl | Base interface for all alarm controls. Contains method declarations specific to alarm devices. | Extends: IJxfsBaseControl |
| Class | JxfsAlarm | Class for alarm control. | Implements: IJxfsAlarmControl |

The following classes and interfaces are used by the J/XFS Alarm Device Service.

| Class or Interface | Name | Description | Extends or Implements |
|---|---|---|---|
| Interface | IJxfsBaseService | Base interface for all services. | |
| Interface | IJxfsAlarmService | Base interface for all alarm services. Contains method declarations specific to alarm devices. | Extends: IJxfsBaseService |

#### 4.1.1.1 Remark on Device Services

The Device Service interface is common for all device services of a specific 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 controlID ). This is always added as the last parameter in every operation.

# 5 Class and Interface Details

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

The Constants, Error Codes, Exceptions, Status Codes and Support classes that are used in the methods are described in special chapters at the end of the documentation.

## 5.1 Exceptions

The methods described for the specific interfaces all can throw at least the following exceptions :

| **Exception** | **Value** |
|---|---|
| *JXFSException* | JXFS_E_CLAIMED |
| | JXFS_E_CLOSED |
| | JXFS_E_INVALID_PARAMETER |
| | JXFS_E_NOT_SUPPORTED |
| | JXFS_E_REMOTE |

Only if a method can throw additional exception this is explicitly mentioned.

## 5.2 IJxfsAlarmControl

An alarm device is a device which is connected to the internal house alarm system. This method enables an "Application" to trigger this device. This would normaly be done in the case of a bank robbery or an unauthorized access. The application can trigger this device even if an application "claim" some where else is pending.

This interface must be implemented by any device control that wants to use security violation signaling. If a device service has not implemented this interface, an exception is thrown.

#### 5.2.1.1 Summary

| Extends | Implements |
|---|---|
| IJxfsBaseControl | |

| Method | Return |
|---|---|
| alarm | int |

### 5.2.2 Methods

#### 5.2.2.1 alarm

| | |
|---|---|
| **Syntax** | ***identificationID alarm( boolean on ) throws JXFSException;*** |
| **Description** | Initiates a device alarm. This method is used to start or stop a notification when a security violation occurred. |
| ***Notice*** | *This method will succeed, even when the device, which incorporates the alarm device, is in the state* CLAIMED. |

**Parameter**

| Type | Name | Meaning |
|---|---|---|
| *boolean* | on | Turns the alarm signaling mechanism on when *true*, otherwise turns it off. |

**Exceptions**

Exceptions can be generated.
When a *alarm* operation is performed on a device service, which has not implemented this interface, a JXFSException is thrown

| Exception | Value |
|---|---|
| *JXFSException* | JXFS_E_ALM_NOT_SUPPORTED |

**Events**

Additional events can be generated.
**OperationCompleteEvent**
When a *alarm* operation is completed an *OperationCompleteEvent* is sent to all registered listeners with following data:

| Field | Value |
|---|---|
| *operationID* | JXFS_O_ALM_ALARM |
| *identificationID* | |
| *result:* | |
| | JXFS_RC_SUCCESSFUL |
| | JXFS_E_ALM_DEVICE_ERROR |
| *data* | |

# 6 Support Classes

No support classes are used by this devices.

## 7 Status Event Classes

There exists no device specific Status Event Classes.

# 8 Codes

## 8.1 Error Codes

| Value | Meaning |
|---|---|
| JXFS_E_ALM_DEVICE_ERROR | The device is malfunctioning or could not initiate the signaling mechanism. |

## 8.2 Exception Codes

| Value | Meaning |
|---|---|
| JXFS_E_ALM_NOT_SUPPORTED | The device does not support alarm signaling. |

## 8.3 Status Codes

No status codes were used.

## 8.4 Operation ID Codes

Following codes specify the operation which generated the OperationCompleteEvent.

#### 8.4.1.1 Alarm device

| Value | Method |
|---|---|
| JXFS_O_ALM_ALARM | *alarm* |