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

WORKSHOP AGREEMENT

CWA 14050-13

November 2000

ICS 35.200; 35.240.40

Extensions for Financial Services (XFS) interface specification - Release 3.0 - Part 13: Alarm Device Class Interface

Ref. No CWA 14050-13:2000 E

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

Table of Contents

Foreword

This CWA is revision 3.0 of the XFS interface specification.

The move from an XFS 2.0 specification (CWA 13449) to a 3.0 specification has been prompted by a series of factors.

Initially, there has been a technical imperative to extend the scope of the existing specification of the XFS Manager to include new devices, such as the Card Embossing Unit.

Similarly, there has also been pressure, through implementation experience and the advance of the Microsoft technology, to extend the functionality and capabilities of the existing devices covered by the specification.

Finally, it is also clear that our customers and the market are asking for an update to a specification, which is now over 2 years old. Increasing market acceptance and the need to meet this demand is driving the Workshop towards this release.

The clear direction of the CEN/ISSS XFS Workshop, therefore, is the delivery of a new Release 3.0 specification based on a C API. It will be delivered with the promise of the protection of technical investment for existing applications and the design to safeguard future developments.

The CEN/ISSS XFS Workshop gathers suppliers 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.

This CWA was formally approved by the XFS Workshop meeting on 2000-10-18. The specification is continuously reviewed and commented in the CEN/ISSS Workshop on XFS. It is therefore expected that an update of the specification will be published in due time as a CWA, superseding this revision 3.0.

The CWA is published as a multi-part document, consisting of:

Part 1: Application Programming Interface (API) - Service Provider Interface (SPI); Programmer's Reference

Part 2: Service Classes Definition; Programmer's Reference

Part 3: Printer Device Class Interface - Programmer's Reference

Part 4: Identification Card Device Class Interface - Programmer's Reference

Part 5: Cash Dispenser Device Class Interface - Programmer's Reference

Part 6: PIN Keypad Device Class Interface - Programmer's Reference

Part 7: Check Reader/Scanner Device Class Interface - Programmer's Reference

Part 8: Depository Device Class Interface - Programmer's Reference

Part 9: Text Terminal Unit Device Class Interface - Programmer's Reference

Part 10: Sensors and Indicators Unit Device Class Interface - Programmer's Reference

Part 11: Vendor Dependent Mode Device Class Interface - Programmer's Reference

Part 12: Camera Device Class Interface - Programmer's Reference

Part 13: Alarm Device Class Interface - Programmer's Reference

Part 14: Card Embossing Unit Class Interface - Programmer's Reference

Part 15: Cash In Module Device Class Interface- Programmer's Reference

Part 16: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 17: Printer Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 18: Identification Card Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 19: Cash Dispenser Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 20: PIN Keypad Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 21: Depository Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 22: Text Terminal Unit Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 23: Sensors and Indicators Unit Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 24: Camera Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 25: Identification Card Device Class Interface - PC/SC Integration Guidelines

In addition to these Programmer's Reference specifications, the reader of this CWA is also referred to a complementary document, called Release Notes. The Release Notes contain clarifications and explanations on the CWA specifications, which are not requiring functional changes. The current version of the Release Notes is available online from http://www.cenorm.be/isss/Workshop/XFS.

Revision History:

3.00	18 October, 2000	Initial release

1. Introduction

1.1 Background to Release 3.0

The CEN XFS Workshop is a continuation of the Banking Solution Vendors Council workshop and maintains a technical commitment to the Win 32 API. However, the XFS Workshop has extended the franchise of multi vendor software by encouraging the participation of both banks and vendors to take part in the deliberations of the creation of an industry standard. This move towards opening the participation beyond the BSVC's original membership has been very succesful with a current membership level of more than 20 companies.

The fundamental aims of the XFS Workshop are to promote a clear and unambiguous specification for both service providers and application developers. This has been achieved to date by sub groups working electronically and quarterly meetings.

The move from an XFS 2.0 specification to a 3.0 specification has been prompted by a series of factors. Initially, there has been a technical imperative to extend the scope of the existing specification of the XFS Manager to include new devices, such as the Card Embossing Unit.

Similarly, there has also been pressure, through implementation experience and the advance of the Microsoft technology, to extend the functionality and capabilities of the existing devices covered by the specification.

Finally, it is also clear that our customers and the market are asking for an update to a specification, which is now over 2 years old. Increasing market acceptance and the need to meet this demand is driving the Workshop towards this release.

The clear direction of the XFS Workshop, therefore, is the delivery of a new Release 3.0 specification based on a C API. It will be delivered with the promise of the protection of technical investment for existing applications and the design to safeguard future developments.

1.2 XFS Service-Specific Programming

The service classes are defined by their service-specific commands and the associated data structures, error codes, messages, etc. These commands are used to request functions that are specific to one or more classes of service providers, but not all of them, and therefore are not included in the common API for basic or administration functions.

When a service-specific command is common among two or more classes of service providers, the syntax of the command is as similar as possible across all services, since a major objective of the Extensions for Financial Services is to standardize function codes and structures for the broadest variety of services. For example, using the **WFSExecute** function, the commands to read data from various services are as similar as possible to each other in their syntax and data structures.

In general, the specific command set for a service class is defined as a superset of the specific capabilities likely to be provided by the developers of the services of that class; thus any particular device will normally support only a subset of the defined command set.
There are three cases in which a service provider may receive a service-specific command that it does not support:

- The requested capability is defined for the class of service providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability is ***not*** considered to be fundamental to the service. In this case, the service provider returns a successful completion, but does no operation. An example would be a request from an application to turn on a control indicator on a passbook printer; the service provider recognizes the command, but since the passbook printer it is managing does not include that indicator, the service provider does no operation and returns a successful completion to the application.
- The requested capability is defined for the class of service providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability ***is*** considered to be fundamental to the service. In this case, a WFS_ERR_UNSUPP_COMMAND error is returned to the calling application. An example would be a request from an application to a cash dispenser to dispense coins; the service provider recognizes the command but, since the cash dispenser it is managing dispenses only notes, returns this error.
- The requested capability is ***not*** defined for the class of service providers by the XFS specification. In this case, a WFS_ERR_INVALID_COMMAND error is returned to the calling application.

This design allows implementation of applications that can be used with a range of services that provide differing subsets of the functionalities that are defined for their service class. Applications may use the **WFSGetInfo** and **WFSAsyncGetInfo** commands to inquire about the capabilities of the service they are about to use, and modify their behavior accordingly, or they may use functions and then deal with WFS_ERR_UNSUPP_COMMAND error returns to make decisions as to how to use the service.

2. Alarms

This specification describes the functionality of the services provided by Alarms (ALM) under XFS, by defining the service-specific commands that can be issued, using the **WFSGetInfo, WFSAsyncGetInfo**, **WFSExecute** and **WFSAsyncExecute** functions. This section describes the functionality of an Alarm (ALM) service that applies to both attended and unattended (self-service) devices.

The Alarm device class is provided as a separate service due to the need to set or reset an Alarm when one or more logical services associated with an attended CDM or unattended (self-service) device are locked. Because logical services can be locked by the application the Alarm is implemented in a separate device class to ensure that a set (trigger) or reset operation can be performed at any time.

The Alarm device class can be part of a compound device, as in the case of many CDMs or self-service terminals, or may be separate physical alarms.

3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer’s Reference, Revision 3.00, October 18, 2000

4. Info Commands

4.1 WFS_INF_ALM_STATUS

Description This command is used to request the Alarm status.

Input Param None.

Output Param `LPWFSALMSTATUS      lpStatus;`

```
typedef struct _wfs_alm_status
   {
   WORD              fwDevice;
   BOOL              bAlarmSet;
   LPSTR             lpszExtra;
   } WFSALMSTATUS, *LPWFSALMSTATUS;
```

fwDevice
Specifies the state of the alarm device as one of the following flags:

Value	Meaning
WFS_ALM_DEVONLINE	The device is present, powered on and online (i.e., operational, not busy processing a request and not in an error state).
WFS_ALM_DEVOFFLINE	The device is offline (e.g., the operator has taken the device offline by turning a switch or pulling out the device).
WFS_ALM_DEVPOWEROFF	The device is powered off or physically not connected.
WFS_ALM_DEVNODEVICE	There is no device intended to be there; e.g. this type of self service machine does not contain such a device or it is internally not configured..
WFS_ALM_DEVUSERERROR	The device is present but a person is preventing proper device operation. The application should suspend the device operation or remove the device from service until the service provider generates a device state change event indicating the condition of the device has changed e.g.the error is removed (WFS_ALM_DEVONLINE) or a permanent error condition has occurred (WFS_ALM_DEVHWERROR).
WFS_ALM_DEVHWERROR	The device is present but inoperable due to a hardware fault that prevents it from being used.
WFS_ALM_DEVBUSY	The device is busy and unable to process an execute command at this time.

bAlarmSet
Specifies the state of the Alarm as either Reset (False) or Set (True).

lpszExtra
Points to a list of vendor-specific, or any other extended information. The information is returned as a series of "*key=value*" strings so that it is easily extensible by service providers. Each string will be null-terminated, with the final string terminating with two null characters.

Error Codes Only the generic error codes defined in [Ref. 1] can be generated by this command.

Comments Applications which require or expect specific information to be present in the *lpszExtra* parameter may not be device or vendor-independent.

4.2 WFS_INF_ALM_CAPABILITIES

Description This command is used to retrieve the capabilities of the Alarm.

Input Param None.

Output Param

```
LPWFSALMCAPS        lpCaps;
```

```
typedef struct _wfs_alm_caps
   {
   WORD         wClass;
   BOOL         bProgrammaticallyDeactivate;
   LPSTR        lpszExtra;
   } WFSALMCAPS, *LPWFSALMCAPS;
```

wClass
Specifies the logical service class. Value is WFS_SERVICE_CLASS_ALM.

bProgrammaticallyDeactivate
Specifies whether the Alarm can be programmatically Deactivated (True) or can not be programmatically Deactivated (False).

lpszExtra
Pointer to a list of vendor-specific, or any other extended information. The information is returned as a series of "*key=value*" strings so that it is easily extensible by service providers. Each string is null-terminated, with the final string terminating with two null characters.

Error Codes Only the generic error codes defined in [Ref. 1] can be generated by this command.

Comments Applications which require or expect specific information to be present in the *lpszExtra* parameter may not be device or vendor-independent.

5. Execute Commands

5.1 WFS_CMD_ALM_SET_ALARM

Description This command is used to trigger an Alarm.

Input Param None

Output Param None.

Error Codes Only the generic error codes defined in [Ref. 1] can be generated by this command.

Events In addition to the generic events defined in [Ref. 1], the following events can be generated as a result of this command:

Value	Meaning
WFS_SRVE_ALM_DEVICE_SET	The alarm device has been triggered.

Comments None**.**

5.2 WFS_CMD_ALM_RESET_ALARM

Description This command is used to reset an Alarm.

Input Param None

Output Param None.

Error Codes Only the generic error codes defined in [Ref. 1] can be generated by this command.

Events In addition to the generic events defined in [Ref. 1], the following events can be generated as a result of this command:

Value	Meaning
WFS_SRVE_ALM_DEVICE_RESET	The alarm device has been reset.

Comments None**.**

5.3 WFS_CMD_ALM_RESET

Description Sends a service reset to the service provider.

Input Param None

Output Param None.

Error Codes Only the generic error codes defined in [Ref. 1] can be generated by this command.

Events Only the generic events defined in [Ref. 1] can be generated by this command.

Comments This command is used by an application control program to cause a device to reset itself to a known good condition.

6. Events

6.1 WFS_SRVE_ALM_DEVICE_SET

Description	The Alarm has been set (triggered) by an external event or a programmatic request to set (trigger) the Alarm.
Event Param	None.
Comments	None**.**

6.2 WFS_SRVE_ALM_DEVICE_RESET

Description	The Alarm has been manually or programmatically reset.
Event Param	None.
Comments	None**.**

7. C - Header file

```
/******************************************************************************
*                                                                             *
* xfsalm.h      XFS – Alarm (ALM) definitions                                 *
*                                                                             *
*               Version 3.00 (10/18/00)                                       *
*                                                                             *
******************************************************************************/

#ifndef __INC_XFSALM__H
#define __INC_XFSALM__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

/* be aware of alignment */
#pragma pack (push, 1)

/* values of WFSALMCAPS.wClass */

#define     WFS_SERVICE_CLASS_ALM               (11)
#define     WFS_SERVICE_CLASS_VERSION_ALM       0x0003
#define     WFS_SERVICE_CLASS_NAME_ALM          "ALM"

#define     ALM_SERVICE_OFFSET                  (WFS_SERVICE_CLASS_ALM * 100)

/* ALM Info Commands */

#define     WFS_INF_ALM_STATUS                  (ALM_SERVICE_OFFSET + 1)
#define     WFS_INF_ALM_CAPABILITIES            (ALM_SERVICE_OFFSET + 2)

/* ALM Execute Commands */

#define     WFS_CMD_ALM_SET_ALARM               (ALM_SERVICE_OFFSET + 1)
#define     WFS_CMD_ALM_RESET_ALARM             (ALM_SERVICE_OFFSET + 2)
#define     WFS_CMD_ALM_RESET                   (ALM_SERVICE_OFFSET + 3)

/* ALM Messages */

#define     WFS_SRVE_ALM_DEVICE_SET             (ALM_SERVICE_OFFSET + 1)
#define     WFS_SRVE_ALM_DEVICE_RESET           (ALM_SERVICE_OFFSET + 2)

/* values of WFSALMSTATUS.fwDevice */
#define     WFS_ALM_DEVONLINE                   WFS_STAT_DEVONLINE
#define     WFS_ALM_DEVOFFLINE                  WFS_STAT_DEVOFFLINE
#define     WFS_ALM_DEVPOWEROFF                 WFS_STAT_DEVPOWEROFF
#define     WFS_ALM_DEVNODEVICE                 WFS_STAT_DEVNODEVICE
#define     WFS_ALM_DEVHWERROR                  WFS_STAT_DEVHWERROR
#define     WFS_ALM_DEVUSERERROR                WFS_STAT_DEVUSERERROR
#define     WFS_ALM_DEVBUSY                     WFS_STAT_DEVBUSY


/*=================================================================*/
/* ALM Info Command Structures */
/*=================================================================*/

typedef struct _wfs_alm_status
{
    WORD     fwDevice;
    BOOL     bAlarmSet;
    LPSTR    lpszExtra;
} WFSALMSTATUS, *LPWFSALMSTATUS;
typedef struct _wfs_alm_caps
{
    WORD     wClass;
    BOOL     bProgrammaticallyDeactivate;
    LPSTR    lpszExtra;
} WFSALMCAPS, * LPWFSALMCAPS;


/* restore alignment */
#pragma pack (pop)

#ifdef __cplusplus
}       /*extern "C"*/
#endif

#endif  /* __INC_XFSALM__H */
```