CEN

WORKSHOP

AGREEMENT

CWA 16926-72

August 2015

ICS 35.240.40; 35.240.15; 35.200

English version

Extensions for Financial Services (XFS) interface specification Release 3.30 - Part 72: Alarm Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

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

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

Ref. No.:CWA 16926-72:2015 E

Table of Contents

European foreword

This CWA is revision 3.30 of the XFS interface specification.

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties on March 19th 2015, the constitution of which was supported by CEN following the public call for participation made on 1998-06-24. 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 3.30.

A list of the individuals and organizations which supported the technical consensus represented by the CEN Workshop Agreement is available from the CEN/XFS Secretariat. The CEN XFS Workshop gathered suppliers as well as banks and other financial service companies.

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 and Scanning 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 Device Class Interface - Programmer's Reference

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

Part 16: Card Dispenser Device Class Interface - Programmer's Reference

Part 17: Barcode Reader Device Class Interface - Programmer's Reference

Part 18: Item Processing Module Device Class Interface- Programmer's Reference

Parts 19 - 28: Reserved for future use.

Parts 29 through 47 constitute an optional addendum to this CWA. They define the integration between the SNMP standard and the set of status and statistical information exported by the Service Providers.

Part 29: XFS MIB Architecture and SNMP Extensions - Programmer's Reference

Part 30: XFS MIB Device Specific Definitions - Printer Device Class

Part 31: XFS MIB Device Specific Definitions - Identification Card Device Class

Part 32: XFS MIB Device Specific Definitions - Cash Dispenser Device Class

Part 33: XFS MIB Device Specific Definitions - PIN Keypad Device Class

Part 34: XFS MIB Device Specific Definitions - Check Reader/Scanner Device Class

Part 35: XFS MIB Device Specific Definitions - Depository Device Class

Part 36: XFS MIB Device Specific Definitions - Text Terminal Unit Device Class

Part 37: XFS MIB Device Specific Definitions - Sensors and Indicators Unit Device Class

Part 38: XFS MIB Device Specific Definitions - Camera Device Class

Part 39: XFS MIB Device Specific Definitions - Alarm Device Class

Part 40: XFS MIB Device Specific Definitions - Card Embossing Unit Class

Part 41: XFS MIB Device Specific Definitions - Cash-In Module Device Class

Part 42: Reserved for future use.

Part 43: XFS MIB Device Specific Definitions - Vendor Dependent Mode Device Class

Part 44: XFS MIB Application Management

Part 45: XFS MIB Device Specific Definitions - Card Dispenser Device Class

Part 46: XFS MIB Device Specific Definitions - Barcode Reader Device Class

Part 47: XFS MIB Device Specific Definitions - Item Processing Module Device Class

Parts 48 - 60 are reserved for future use.

Part 61: Application Programming Interface (API) - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Service Provider Interface (SPI) - Programmer's Reference

Part 62: Printer and Scanning Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 63: Identification Card Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 65: PIN Keypad Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 67: Depository Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 71: Camera Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 72: Alarm Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 74: Cash-In Module Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 75: Card Dispenser Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 76: Barcode Reader Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 77: Item Processing Module Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

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.cen.eu/work/areas/ict/ebusiness/pages/ws-xfs.aspx.

The information in this document represents the Workshop's current views on the issues discussed as of the date of publication. It is furnished for informational purposes only and is subject to change without notice. CEN makes no warranty, express or implied, with respect to this document.

The formal process followed by the Workshop in the development of the CEN Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN-CENELEC Management Centre can be held accountable for the technical content of the CEN Workshop Agreement or possible conflict with standards or legislation. This CEN Workshop Agreement can in no way be held as being an official standard developed by CEN and its members.

The final review/endorsement round for this CWA was started on 2015-01-16 and was successfully closed on 2015-03-19. The final text of this CWA was submitted to CEN for publication on 2015-06-19. The specification is continuously reviewed and commented in the CEN 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 3.30.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN [and/or CENELEC] shall not be held responsible for identifying any or all such patent rights.

According to the CEN-CENELEC Internal Regulations, the national standards organizations of the following countries are bound to implement this European Standard: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

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

1. Migration Information

XFS 3.30 has been designed to minimize backwards compatibility issues. This document highlights the changes made to the ALM device class between version 3.20 and 3.30, by highlighting the additions and deletions to the text.

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.

3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer's Reference Revision 3.~~20~~30

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;
     WORD                  wAntiFraudModule;
     } 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.
WFS_ALM_DEVFRAUDATTEMPT	The device is present but is inoperable because it has detected a fraud attempt.
WFS_ALM_DEVPOTENTIALFRAUD	The device has detected a potential fraud attempt and is capable of remaining in service. In this case the application should make the decision as to whether to take the device offline.

bAlarmSet
Specifies the state of the Alarm as either Reset (FALSE) or Set (TRUE).

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. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.

wAntiFraudModule
Specifies the state of the anti-fraud module as one of the following values:

Value	Meaning
WFS_ALM_AFMNOTSUPP	No anti-fraud module is available.
WFS_ALM_AFMOK	Anti-fraud module is in a good state and no foreign device is detected.
WFS_ALM_AFMINOP	Anti-fraud module is inoperable.
WFS_ALM_AFMDEVICEDETECTED	Anti-fraud module detected the presence of a foreign device.
WFS_ALM_AFMUNKNOWN	The state of the anti-fraud module cannot be determined.

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.

In the case where communications with the device has been lost, the *fwDevice* field will report WFS_ALM_DEVPOWEROFF when the device has been removed or WFS_ALM_DEVHWERROR if the communications are unexpectedly lost. All other fields should contain a value based on the following rules and priority:

1. Report the value as unknown.
2. Report the value as a general h/w error.
3. Report the value as the last known value.

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;
     BOOL                  bAntiFraudModule;
     LPDWORD               lpdwSynchronizableCommands;
     } WFSALMCAPS, *LPWFSALMCAPS;
```

wClass
Specifies the logical service class as 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. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.

bAntiFraudModule
Specifies whether the anti-fraud module is available. This can either be TRUE if available or FALSE if not available.

lpdwSynchronizableCommands
Pointer to a zero-terminated list of DWORDs which contains the execute command IDs that can be synchronized. If no execute command can be synchronized then this parameter will be NULL.

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.

5.4 WFS_CMD_ALM_SYNCHRONIZE_COMMAND

Description This command is used to reduce response time of a command (e.g. for synchronization with display) as well as to synchronize actions of the different device classes. This command is intended to be used only on hardware which is capable of synchronizing functionality within a single device class or with other device classes.

The list of execute commands which this command supports for synchronization is retrieved in the *lpdwSynchronizableCommands* parameter of the WFS_INF_ALM_CAPABILITIES.

This command is optional, i.e. any other command can be called without having to call it in advance. Any preparation that occurs by calling this command will not affect any other subsequent command. However, any subsequent execute command other than the one that was specified in the *dwCommand* input parameter will execute normally and may invalidate the pending synchronization. In this case the application should call the WFS_CMD_ALM_SYNCHRONIZE_COMMAND again in order to start a synchronization.

Input Param LPWFSALMSYNCHRONIZECOMMAND lpSynchronizeCommand;

```
typedef struct _wfs_alm_synchronize_command
    {
    DWORD                   dwCommand;
    LPVOID                  lpCmdData;
    } WFSALMSYNCHRONIZECOMMAND, *LPWFSALMSYNCHRONIZECOMMAND;
```

dwCommand
The command ID of the command to be synchronized and executed next.

lpCmdData
Pointer to data or a data structure that represents the parameter that is normally associated with the command that is specified in *dwCommand*. This parameter can be NULL if no command input parameter is needed or if this detail is not needed to synchronize for the command.

It will be device-dependent whether the synchronization is effective or not in the case where the application synchronizes for a command with this command specifying a parameter but subsequently executes the synchronized command with a different parameter. This case should not result in an error; however, the preparation effect could be different from what the application expects. The application should, therefore, make sure to use the same parameter between *lpCmdData* of this command and the subsequent corresponding execute command.

Output Param None.

Error Codes In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

Value	Meaning
WFS_ERR_ALM_COMMANDUNSUPP	The command specified in the *dwCommand* field is not supported by the Service Provider.
WFS_ERR_ALM_SYNCHRONIZEUNSUPP	The preparation for the command specified in the *dwCommand* with the parameter specified in the *lpCmdData* is not supported by the Service Provider.

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

Comments For sample flows of this synchronization see the [Ref 1] Appendix C.

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 3.2030  (March 02 201119 2015)                        *
*                                                                             *
*                                                                             *
******************************************************************************/

#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       0x14030x1E03 /*Version 3.2030 */
#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)
#define     WFS_CMD_ALM_SYNCHRONIZE_COMMAND     (ALM_SERVICE_OFFSET + 4)

/* 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
#define     WFS_ALM_DEVFRAUDATTEMPT             WFS_STAT_DEVFRAUDATTEMPT
#define     WFS_ALM_DEVPOTENTIALFRAUD           WFS_STAT_DEVPOTENTIALFRAUD

/* values of WFSALMSTATUS.wAntiFraudModule */

#define     WFS_ALM_AFMNOTSUPP                  (0)
#define     WFS_ALM_AFMOK                       (1)
#define     WFS_ALM_AFMINOP                     (2)
#define     WFS_ALM_AFMDEVICEDETECTED           (3)
#define     WFS_ALM_AFMUNKNOWN                  (4)

/* XFS ALM Errors */
```

```
#define WFS_ERR_ALM_COMMANDUNSUPP                (-(ALM_SERVICE_OFFSET + 0))
#define WFS_ERR_ALM_SYNCHRONIZEUNSUPP            (-(ALM_SERVICE_OFFSET + 1))

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

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

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

typedef struct _wfs_alm_synchronize_command
{
    DWORD                   dwCommand;
    LPVOID                  lpCmdData;
} WFSALMSYNCHRONIZECOMMAND, *LPWFSALMSYNCHRONIZECOMMAND;

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

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