CEN

WORKSHOP

AGREEMENT

CWA 13449-12

December 1998

ICS 35.160.40;35.200;35.240.40

English version

Extensions for Financial Services (XFS) interface specification - Part 12: Camera Device Class Interface - Programmer's Interface

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties, the constitution of which is indicated in the foreword of this Workshop Agreement.

The formal process followed by the Workshop in the development of this Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN Central Secretariat can be held accountable for the technical content of this CEN Workshop Agreement or possible conflicts 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.

This CEN Workshop Agreement is publicly available as a reference document from the CEN Members National Standard Bodies.

CEN Members are the National Standards Bodies of Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

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

Central Secretariat: rue de Stassart, 36 B-1050 Brussels

Ref. No. CWA 13449-12:1998 E

Contents

Foreword

This CWA is revision 2.0 of the XFS interface specification. Release 2.0 extends the scope of the XFS interface specification to include both the self service/ATM environment as well as the branch environment. The new specification now fully supports cameras, deposit units, identification cards, PIN pads, sensors and indicator units, text terminals, cash dispenser modules and a wide variety of printing mechanisms.

This specification was originally developed by the Banking Solutions Vendor Council (BSVC), and is endorsed by the CEN/ISSS Workshop on XFS. This Workshop gathers both suppliers (among others the BSVC members) 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 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 2.00.

This CWA is supplemented by a set of release notes, which are available from the CEN/ISSS Secretariat (an on-line version of these release notes is available from http://www.cenorm.be/isss/Workshop/XFS/release-notes.htm).

0. Introduction

This is part 12 of the multi-part CWA 13449, describing Release 2.0 of the XFS interface specification.

The full CWA 13449 "Extensions for Financial Services (XFS) interface specification"consists of the following parts:

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

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 from the CEN/ISSS Secretariat (contact isss@cenorm.be or download from http://www.cenorm.be/isss/ Workshop/XFS/release-notes.htm).

The information in this document originally contributed by members of the Banking Solutions Vendor Council and endorsed by the CEN/ISSS Workshop on XFS, 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/ISSS makes no warranty, express or implied, with respect to this document.

The XFS specifications are now further developed in the CEN/ISSS Workshop on XFS. 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).

A Software Development Kit (SDK) which supplies the components and tools to allow the implementation of compliant applications and services is available from Microsoft[1].

To the extent that date processing occurs, all XFS Workshop participants agree that the XFS specifications are Year 2000 compliant.

Revision History:

1.0	**May 24, 1993**	**Initial release of API and SPI specification**
1.11	**February 3, 1995**	**Separation of specification into separate documents for API/SPI and service class definitions, with updates**
2.00	**November 11, 1996**	**Updated release encompassing self-service environment.**
	October 6, 1998	**WOSA/XFS Release 2.00 as originally developed by the BSVC, has been formally accepted as a CEN Workshop Agreement by the CEN/ISSS XFS Workshop and the name WOSA/XFS has been changed into XFS. In spite of the name change, certain occurrencies of WOSA/XFS however still appear in the documentation, for compatibility reasons**

[1] Microsoft is a registered trademark, and Windows and Windows NT are trademarks of Microsoft Corporation

1. 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 specification is to standardize command 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 the union 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_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. Banking Cameras

This specification describes the functionality of the services provided by the Camera (CAM) services under XFS, by defining the service-specific commands that can be issued, using the **WFSGetInfo, WFSAsyncGetInfo**, **WFSExecute** and **WFSAsyncExecute** functions.

Banking camera systems usually consist of a recorder, a video mixer and one or more cameras. If there are several cameras, each camera focuses a special place within the self-service area (eg. the room, the customer or the cash tray). By using the video mixer it can be decided, which of the cameras should take the next photo. Furthermore data can be given to be inserted in the photo (eg. date, time or bankcode).

If there is only one camera that can switch to take photos from different positions, it is presented by the service provider as a set of cameras, one for each of its possible positions.

3. Info Commands

3.1 WFS_INF_CAM_STATUS

Description This command reports the full range of information available, including the information that is provided by the service provider.

Input Param None.

Output Param

```
LPWFSCAMSTATUS      lpStatus;

typedef struct _wfs_cam_status
   {
   WORD          fwDevice;
   WORD          fwMedia;
   WORD          fwCameras[WFS_CAM_CAMERAS_SIZE];
   USHORT        usPictures;
   LPSTR         lpszExtra;
   } WFSCAMSTATUS, * LPWFSCAMSTATUS;
```

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

Value	Meaning
WFS_CAM_DEVONLINE	The device is on-line. The device is present and operational (i.e. not busy processing a request and does not have a hardware error).
WFS_CAM_DEVOFFLINE	The device is off-line. The device is present and powered on but it is not operational (e.g. a switch may have been used to change it to an off-line state).
WFS_CAM_DEVPOWEROFF	The device is powered off. The device is present, but is currently powered off.
WFS_CAM_DEVBUSY	The device is busy processing a request. The device is present and an EXECUTE request is currently being processed.
WFS_CAM_DEVNODEVICE	There is no device connected.
WFS_CAM_DEVHWERROR	The device is inoperable due to a hardware error. The device is present but a hardware fault prevents it from being used.
WFS_CAM_DEVUSERERROR	The device is present but a person is preventing proper operation. The application should suspend the device operation or remove the device from service until the service provider generates a device state change event indicating the condition of the device has changed i.e. the error is removed (WFS_CAM_DEVONLINE) or a permanent error condition has occurred (WFS_CAM_DEVHWERROR).

fwMedia
Specifies the state of the recording media. Specified as one of the following flags:

Value	Meaning
WFS_CAM_MEDIAOK	The media is in a good state.
WFS_CAM_MEDIAHIGH	The media is almost full (threshold).
WFS_CAM_MEDIAFULL	The media is full.
WFS_CAM_MEDIAUNKNOWN	Due to a hardware error or other condition, the state of the media cannot be determined.

fwCameras[...]
Specifies the state of the cameras. A number of cameras are defined below. The maximum camera index is WFS_CAM_CAMERAS_MAX.

fwCameras[WFS_CAM_ROOM]
Specifies the state of the camera that monitors the whole self-service area. Specified as one of the following flags:

Value	Meaning
WFS_CAM_CAMNOTSUPP	The camera is not supported.
WFS_CAM_CAMOK	The camera is in a good state.
WFS_CAM_CAMINOP	The camera is inoperative.
WFS_CAM_CAMUNKNOWN	Due to a hardware error or other condition, the state of the camera cannot be determined.

fwCameras[WFS_CAM_PERSON]
Specifies the state of the camera that monitors the person standing in front of the self-service machine. Specified as one of the following flags:

Value	Meaning
WFS_CAM_CAMNOTSUPP	The camera is not supported.
WFS_CAM_CAMOK	The camera is in a good state.
WFS_CAM_CAMINOP	The camera is inoperative.
WFS_CAM_CAMUNKNOWN	Due to a hardware error or other condition, the state of the camera cannot be determined.

fwCameras[WFS_CAM_EXITSLOT]
Specifies the state of the camera that monitors the exit slot(s) of the self-service machine. Specified as one of the following flags:

Value	Meaning
WFS_CAM_CAMNOTSUPP	The camera is not supported.
WFS_CAM_CAMOK	The camera is in a good state.
WFS_CAM_CAMINOP	The camera is inoperative.
WFS_CAM_CAMUNKNOWN	Due to a hardware error or other condition, the state of the camera cannot be determined.

usPictures
Specifies the number of pictures stored on the recording media.

lpszExtra
Specifies 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 There are no additional error codes 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.

3.2 WFS_INF_CAM_CAPABILITIES

Description This command is used to retrieve the capabilities of the Camera System

Input Param None.

Output Param

```
LPWFSCAMCAPS  lpCaps;

typedef struct _wfs_cam_caps
   {
   WORD          wClass;
   WORD          fwType;
   WORD          fwCameras[WFS_CAM_CAMERAS_SIZE];
   USHORT        usMaxPictures;
   WORD          fwCamData;
   USHORT        usMaxDataLength;
   LPSTR         lpszExtra;
   } WFSCAMCAPS, * LPWFSCAMCAPS;
```

wClass
Specifies the logical service class, value is:
WFS_SERVICE_CLASS_CAM

fwType
Specifies the type of the camera device; only current value is:

Value	Meaning
WFS_CAM_TYPE_CAM	Camera system

fwCameras[...]
Specifies which cameras are available. A number of cameras are defined below. The maximum camera index is WFS_CAM_CAMERAS_MAX.

fwCameras[WFS_CAM_ROOM]
Specifies whether the camera that monitors the whole self-service area is available. Specified as one of the following flags:

Value	Meaning
WFS_CAM_NOT_AVAILABLE	This camera is not available.
WFS_CAM_AVAILABLE	This camera is available.

fwCameras[WFS_CAM_PERSON]
Specifies whether the camera that monitors the person standing in front of the self-service machine is available. Specified as one of the following flags:

Value	Meaning
WFS_CAM_NOT_AVAILABLE	This camera is not available.
WFS_CAM_AVAILABLE	This camera is available.

fwCameras[WFS_CAM_EXITSLOT]
Specifies whether the camera that monitors the exit slot(s) of the self-service machine is available. Specified as one of the following flags:

Value	Meaning
WFS_CAM_NOT_AVAILABLE	This camera is not available.
WFS_CAM_AVAILABLE	This camera is available.

usMaxPictures
Specifies the maximum number of pictures that can be stored on the recording media.

fwCamData
Specifies, if data can be added to the picture.

Specified as a combination of the following flags:

Value	Meaning
WFS_CAM_NOTADD	No data can be added to the picture.
WFS_CAM_AUTOADD	Data is added automatically to the picture.
WFS_CAM_MANADD	Data can be added manually to the picture using the filed lpszCamData in the WFS_CMD_CAM_TAKE_PICTURE command.

usMaxDataLength
Specifies the maximum length of the data that is displayed on the photo. Zero, if data cannot be manually added to the picture.

lpszExtra
Specifies 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 There are no additional error codes 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. Execute Commands

4.1 WFS_CMD_CAM_TAKE_PICTURE

Description This command is used to start the recording of the camera system. It is possible to select which camera or which camera position should be used to take a picture. Furthermore data can be sent to be displayed on the photo.

Input Param

```
LPWFSCAMTAKEPICT    lpTakePict;

typedef struct _wfs_cam_take_picture
   {
   WORD          wCamera;
   LPSTR         lpszCamData;
   } WFSCAMTAKEPICT, * LPWFSCAMTAKEPICT;
```

wCamera
Specifies the camera that should take the photo as one of the following flags:

Value	Meaning
WFS_CAM_ROOM	Monitors the whole self-service area.
WFS_CAM_PERSON	Monitors the person standing in front of the self-service machine.
WFS_CAM_EXITSLOT	Monitors the exit slot(s) of the self-service machine.

lpszCamData
Specifies the text string to be displayed on the photo. If the maximum text length is exceeded, it will be truncated. In this case or if the text given is invalid an execute event WFS_EXEE_CAM_INVALIDDATA is generated. Nevertheless the picture is taken.

Output Param None.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_CAM_CAMNOTSUPP	The specified camera is not supported.
WFS_ERR_CAM_CAMINOP	The specified camera is inoperable.
WFS_ERR_CAM_MEDIAFULL	The recording media is full.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_USRE_CAM_MEDIATHRESHOLD	The state of the recording media reached a threshold.
WFS_EXEE_CAM_INVALIDDATA	The text string given is to long or in some other way invalid.

Comments None.

5. Events

5.1 WFS_USRE_CAM_MEDIATHRESHOLD

Description This user event is used to specify that the state of the recording media reached a threshold.

Event Param LPWORD lpwMediaThreshold;

Specified as one of the following flags:

Value	Meaning
WFS_CAM_MEDIAHIGH	The recording media is almost full.
WFS_CAM_MEDIAFULL	The recording media is full.

Comments None.

5.2 WFS_EXEE_CAM_INVALIDDATA

Description This execute event is used to specify that the text string given was to long or in some other way invalid.

Event Param None.

Comments None.

6. C - Header file

```
/******************************************************************************
*                                                                             *
* xfscam.h      XFS - Camera (CAM) definitions                                *
*                                                                             *
*               Version 2.00 (11/11/96)                                       *
*                                                                             *
******************************************************************************/

#ifndef __INC_XFSCAM__H
#define __INC_XFSCAM__H

#ifdef __cplusplus
extern "C" {
#endif

#include    <xfsapi.h>

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

/* values of WFSCAMCAPS.wClass */

#define     WFS_SERVICE_CLASS_CAM               (10)
#define     WFS_SERVICE_VERSION_CAM             (0x0002) /* Version 2.00 */
#define     WFS_SERVICE_NAME_CAM                "CAM"

#define     CAM_SERVICE_OFFSET                  (WFS_SERVICE_CLASS_CAM * 100)

/* CAM Info Commands */

#define     WFS_INF_CAM_STATUS                  (CAM_SERVICE_OFFSET + 1)
#define     WFS_INF_CAM_CAPABILITIES            (CAM_SERVICE_OFFSET + 2)

/* CAM Execute Commands */

#define     WFS_CMD_CAM_TAKE_PICTURE            (CAM_SERVICE_OFFSET + 1)

/* CAM Messages */

#define     WFS_USRE_CAM_MEDIATHRESHOLD         (CAM_SERVICE_OFFSET + 1)
#define     WFS_EXEE_CAM_INVALIDDATA            (CAM_SERVICE_OFFSET + 2)

/* values of WFSCAMSTATUS.fwDevice */

#define     WFS_CAM_DEVONLINE                   WFS_STAT_DEVONLINE
#define     WFS_CAM_DEVOFFLINE                  WFS_STAT_DEVOFFLINE
#define     WFS_CAM_DEVPOWEROFF                 WFS_STAT_DEVPOWEROFF
#define     WFS_CAM_DEVBUSY                     WFS_STAT_DEVBUSY
#define     WFS_CAM_DEVNODEVICE                 WFS_STAT_DEVNODEVICE
#define     WFS_CAM_DEVHWERROR                  WFS_STAT_DEVHWERROR
#define     WFS_CAM_DEVUSERERROR                WFS_STAT_DEVUSERERROR

/* number of cameras supported/length of WFSCAMSTATUS.fwCameras field */

#define     WFS_CAM_CAMERAS_SIZE                (8)
#define     WFS_CAM_CAMERAS_MAX                 (WFS_CAM_CAMERAS_SIZE - 1)

/* indices of WFSCAMSTATUS.fwCameras [...]
              WFSCAMCAPS.fwCameras [...]
              WFSCAMTAKEPICT.wCamera             */
#define     WFS_CAM_ROOM                        (0)
#define     WFS_CAM_PERSON                      (1)
#define     WFS_CAM_EXITSLOT                    (2)

/* values of WFSCAMSTATUS.fwMedia */

#define     WFS_CAM_MEDIAOK                     (0)
#define     WFS_CAM_MEDIAHIGH                   (1)
#define     WFS_CAM_MEDIAFULL                   (2)
```

```
#define     WFS_CAM_MEDIAUNKNOWN                (3)

/* values of WFSCAMSTATUS.fwCameras */

#define     WFS_CAM_CAMNOTSUPP                  (0)
#define     WFS_CAM_CAMOK                       (1)
#define     WFS_CAM_CAMINOP                     (2)
#define     WFS_CAM_CAMUNKNOWN                  (3)

/* values of WFSCAMCAPS.fwType */

#define     WFS_CAM_TYPE_CAM                    (1)


/* values of WFSCAMCAPS.fwCamData */

#define     WFS_CAM_NOTADD                      (0)
#define     WFS_CAM_AUTOADD                     (1)
#define     WFS_CAM_MANADD                      (2)


/* XFS CAM Errors */

#define WFS_ERR_CAM_CAMNOTSUPP          (-(CAM_SERVICE_OFFSET + 0))
#define WFS_ERR_CAM_MEDIAFULL           (-(CAM_SERVICE_OFFSET + 1))
#define WFS_ERR_CAM_CAMINOP             (-(CAM_SERVICE_OFFSET + 2))


/*=================================================================*/
/* CAM Info Command Structures */
/*=================================================================*/

typedef struct _wfs_cam_status
{
    WORD            fwDevice;
    WORD            fwMedia;
    WORD            fwCameras[WFS_CAM_CAMERAS_SIZE];
    USHORT          usPictures;
    LPSTR           lpszExtra;
} WFSCAMSTATUS, *LPWFSCAMSTATUS;

typedef struct _wfs_cam_caps
{
    WORD            wClass;
    WORD            fwType;
    WORD            fwCameras[WFS_CAM_CAMERAS_SIZE];
    USHORT          usMaxPictures;
    WORD            fwCamData;
    USHORT          usMaxDataLength;
    LPSTR           lpszExtra;
} WFSCAMCAPS, * LPWFSCAMCAPS;


/*=================================================================*/
/* CAM Execute Command Structures */
/*=================================================================*/

typedef struct _wfs_cam_take_picture
{
    WORD            wCamera;
    LPSTR           lpszCamData;
} WFSCAMTAKEPICT, *LPWFSCAMTAKEPICT;

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

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

#endif  /* __INC_XFSCAM__H */
```