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

WORKSHOP AGREEMENT

CWA 14050-10

March 2002

ICS 35.200; 35.240.40

Supersedes CWA 14050-10:2000

Extensions for Financial Services (XFS) interface specification - Release 3.01 - Part 10: Sensors and Indicators Unit Device Class Interface

Ref. No CWA 14050-10:2002 E

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

Table of Contents

page

Foreword

This CWA is revision 3.01 of the XFS interface specification.

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 2001-11-16. 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 3.01.

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.

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/ISSS makes no warranty, express or implied, with respect to this document.

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
2.00	November 11, 1996	Update release encompassing the self-service environment
3.00	October 18, 2000	Addition of the reset command. • Addition of the auxiliaries WFS_SIU_REMOTE_STATUS_MONITOR and WFS_SIU_AUDIBLE_ALARM • Addition of WFS_SIU_SCANNER, WFS_SIU_DOCUMENTPRINTER and WFS_SIU_COINACCEPTOR guidance lights. For a detailed description see CWA 14050-23 SIU Migration from Version 2.00 to Version 3.00, Revision 1.00, October 18, 2000.
3.01	November 16, 2001	Addition of an enhanced audio device. Required for support of American Disabilities Act.

1. Introduction

1.1 Background to Release 3.01

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 3.01 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 3.01 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 WOSA/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 WOSA Extensions for Financial Services 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 WOSA/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 WOSA/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 WOSA/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. Sensors and Indicators Unit

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

This section describes the functions provided by a generic Sensors and Indicators Unit service. This service allows for the operation of the following categories of ports:

- Door sensors, such as cabinet, safe or vandal shield doors;
- Alarm sensors, such as tamper, seismic or heat sensors;
- Generic sensors, such as proximity or ambient light sensors;
- Key switch sensors, such as the ATM operator switch;
- Lamp/sign indicators, such as fascia light or audio indicators;
- Auxiliary indicators.
- Audio jack device, for use by the partially sighted.

In self-service devices, the sensors and indicators unit is capable of dealing with external sensors, such as door switches, locks, alarms and proximity sensors, as well as external indicators, such as turning on lamps or heating.

2.1 Audio Jack Overview

The Audio Jack device is provided to support the requirements of the American Disabilities Act. This device allows audio feedback publicly and / or via the consumers' personal headset (vendor hardware permitting). For privacy, the device allows input to only be directed to the consumers' headset. In 'auto' & 'semi-auto' mode (and where the vendor's hardware allows), public transmission of audio can be automatically inhibited when the consumer's headset is plugged in to the audio jack. In 'auto' mode (and where the vendor's hardware allows), public transmission of audio can be automatically re-activated when the consumer's headset is unplugged from the audio jack

The audio jack provides the application with the following information

- If the headset is present
- Whether the audio output is to the speakers or headset
- Privacy\public mode: ie. Whether insertion of a headset automatically switches public audio on or off.

The device is managed by a sensor WFS_SIU_ENHANCEDAUDIO, and an auxiliary WFS_SIU_ENHANCEDAUDIOCONTROL.

The WFS_SIU_ENHANCEDAUDIO sensor is used to

- provide information on the presence of the Audio Jack device
- to report whether a headset is currently attached
- report state change events when a headset is inserted or removed.

The WFS_SIU_ENHANCEDAUDIOCONTROL auxiliary is used to control the behaviour of the Audio Jack. It allows the application to,

- set the mode of the Audio Jack – auto mode, semi-auto mode or manual mode.
- Set the state of the Audio Jack – public or private.

There are no events associated with this auxiliary.

A full description of auto, semi-auto & manual mode, as well as public & private states is contained in the following pages.

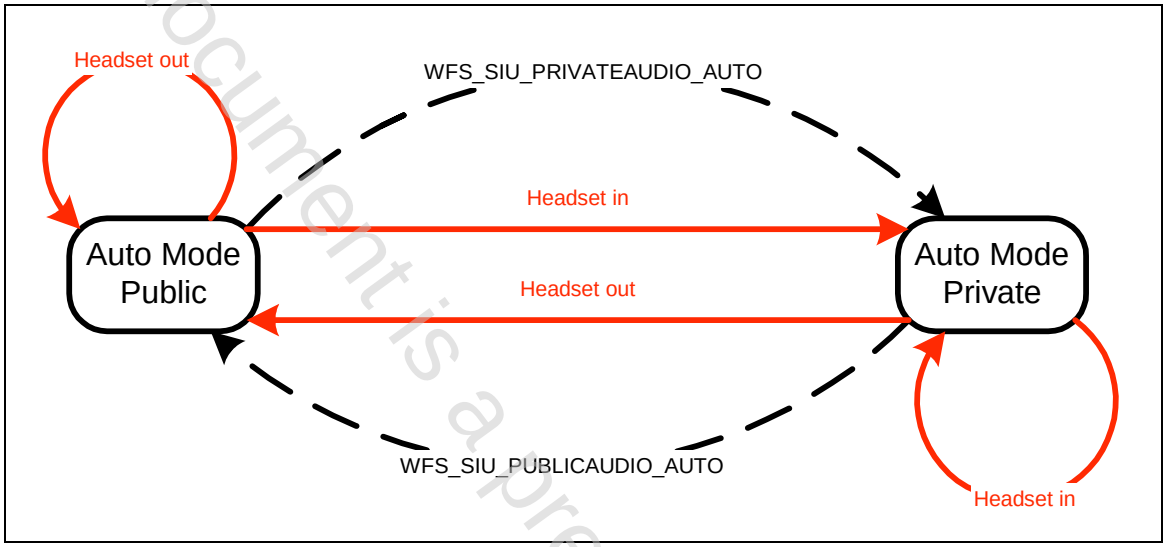
The following describes the device behaviour during auto and manual mode.

Auto Mode

In auto mode, when a consumer headset is plugged into the jack, the audio is automatically directed to the headset and the audio is no longer sent to the speakers. When the headset is removed the audio is redirected to the speakers. The following state diagram completely describes the behaviour of the device in auto mode

State Description

Auto Mode Public	audio output is played through the public speakers only
Auto Mode Private	audio is played through the consumer headset only



Auto-mode State diagram 1

The dashed-line transitions are caused by application calls to WFS_CMD_SIU_SET_PORT or WFS_CMD_SIU_SET_AUXILIARY for the WFS_SIU_ENHANCEDAUDIOCONTROL auxiliary with values of WFS_SIU_PRIVATEAUDIO_AUTO or WFS_SIU_PUBLICAUDIO_AUTO

Note that some vendor implementations may not be have the ability to allow the application to command the service provider to transition between public and private states. To determine if this feature is available, the application can query the field fwAuxiliaries[WFS_SIU_ENHANCEDAUDIOCONTROL] in the WFSSIUCAPS structure.

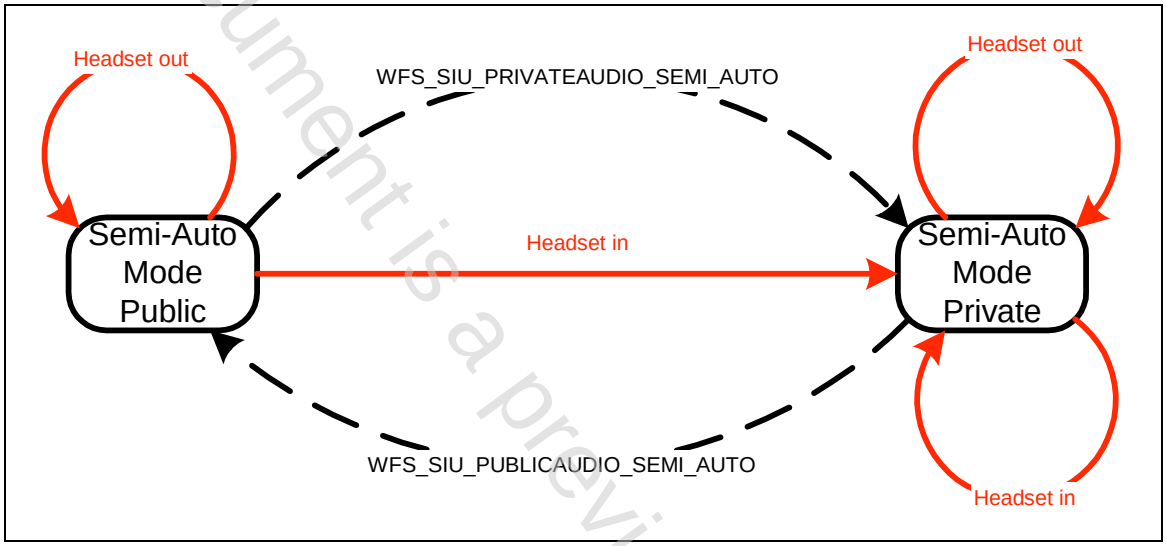
Semi-Auto Mode

This mode is required to ensure customer sensitive information is not broadcast via the public speakers when the consumer's headset is deliberately or otherwise unplugged.

In semi-auto mode, when a consumer headset is plugged into the jack, the audio is automatically directed to the headset and the audio is no longer sent to the speakers. When the headset is removed the audio remains via the jack. If required, the application must explicitly return the device to its public state if audio is required via the speakers. The following state diagram completely describes the behaviour of the device in auto mode

State Description

Semi-Auto Mode Public	audio output is played through the public speakers only
Semi-Auto Mode Private	audio is played through the consumer headset only



Semi-Auto-mode State diagram 2

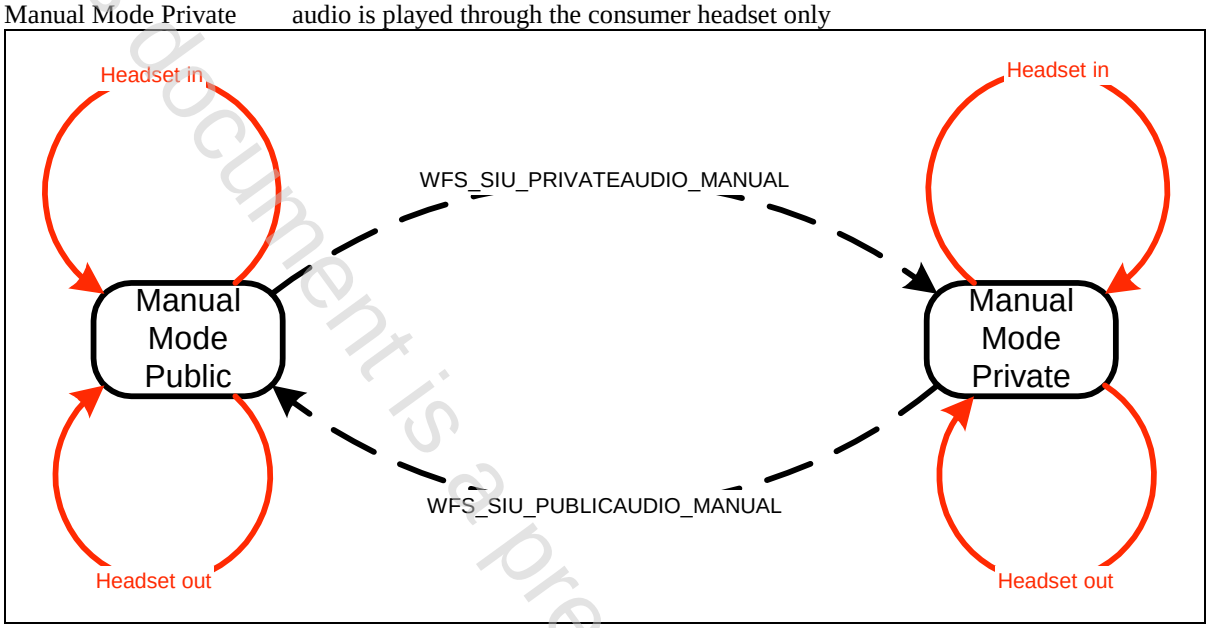
The dashed-line transitions are caused by application calls to WFS_CMD_SIU_SET_PORT or WFS_CMD_SIU_SET_AUXILIARY for the WFS_SIU_ENHANCEDAUDIOCONTROL auxiliary with values of WFS_SIU_PRIVATEAUDIO_AUTO or WFS_SIU_PUBLICAUDIO_AUTO

Manual mode

In manual mode, when a consumer headset is plugged into the jack, the audio remains directed at the existing interface (i.e. the speaker), The application must explicitly change to the other mode, if required. Note that the application must explicitly return the device to its public state if audio is required via the speakers. The following state diagram completely describes the behaviour of the device in manual mode

State Description

Manual Mode Public	audio output is played through the public speakers only
Manual Mode Private	audio is played through the consumer headset only



Manual Mode State Diagram 1

The dashed-line transitions are caused by application calls to WFS_CMD_SIU_SET_PORT or WFS_CMD_SIU_SET_AUXILIARY for the WFS_SIU_ENHANCEDAUDIOCONTROL auxiliary with values of WFS_SIU_PRIVATEAUDIO_MANUAL or WFS_SIU_PUBLICAUDIO_MANUAL

Inte-Mode behaviour

The values described in the previous sections (_AUTO, _SEMI_AUTO, and _MANUAL, etc) can also be used to move from one mode to another. This will then change the mode of the device.

Notes

- Note that if a vendor device does not support auto-mode, or semi-auto mode then the WFS_EXEE_SIU_PORT_ERROR event is received on any attempt to call WFS_CMD_SIU_SET_PORT, etc with the WFS_SIU_PUBLICAUDIO_AUTO, WFS_PRIVATEAUDIO_AUTO, WFS_SIU_PUBLICAUDIO_SEMI_AUTO, and WFS_PRIVATEAUDIO_SEMI_AUTO settings. The same event is generated if calls to change the mode to manual are received when the vendor device does not support manual mode.
- The existing *WFS_SIU_VOLUME* auxiliary can be used to control the volume setting of any audio delivered to connected headset, as well as the speakers. Independent volume control of the speakers and headset is not supported.
- Any 'beep' tones generated by the PINPAD, etc will be fed to a connected headset (vendor hardware permitting).

3. References

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