CEN

WORKSHOP

AGREEMENT

CWA 15748-10

July 2008

ICS 35.240.50

Supersedes CWA 15748-10:2008, February

English version

Extensions for Financial Services (XFS) interface specification - Release 3.10 - Part 10: Sensors and Indicators Unit Device Class Interface - Programmer's Reference

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

Management Centre: rue de Stassart, 36   B-1050 Brussels

Ref. No.:CWA 15748-10:2008 D/E/F

Table of Contents

Foreword

This CWA is revision 3.10 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 2007-11-29. 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 3.10.

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) - Service Provider Interface (SPI) - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 62: Printer Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 63: Identification Card Device Class Interface - Migration from Version 3.02 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 65: PIN Keypad Device Class Interface - Migration from Version 3.03 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 67: Depository Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.01 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 71: Camera Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 72: Alarm Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 74: Cash-In Module Device Class Interface - Migration from Version 3.02 (CWA 14050) to Version 3.10 (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/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.

This CEN Workshop Agreement is publicly available as a reference document from the National Members of CEN : AENOR, AFNOR, ASRO, BDS, BSI, CSNI, CYS, DIN, DS, ELOT, EVS, IBN, IPQ, IST, LVS, LST, MSA, MSZT, NEN, NSAI, ON, PKN, SEE, SIS, SIST, SFS, SN, SNV, SUTN and UNI.

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

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.
2.0	November 11, 1996	Update release encompassing the self-service environment
3.0	October 18, 2000	Updated as follows 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 2.0 to Version 3.0, October 18, 2000.
3.01	November 16, 2001	Addition of an enhanced audio device. Required for support of American Disabilities Act.
3.10	November 29, 2007	For a description of changes see CWA 15748-69:2007 SIU Migration from Version 3.01 (see CWA 14050) to Version 3.10.

1. Introduction

1.1 Background to Release 3.10

The CEN/ISSS XFS Workshop aims to promote a clear and unambiguous specification defining a multi-vendor software interface to financial peripheral devices. The XFS (eXtensions for Financial Services) specifications are developed within the CEN/ISSS (European Committee for Standardization/Information Society Standardization System) Workshop environment. CEN/ISSS Workshops aim to arrive at a European consensus on an issue that can be published as a CEN Workshop Agreement (CWA).

The CEN/ISSS XFS Workshop encourages the participation of both banks and vendors in the deliberations required to create an industry standard. The CEN/ISSS XFS Workshop achieves its goals by focused sub-groups working electronically and meeting quarterly.

Release 3.10 of the XFS specification is based on a C API and is delivered with the continued promise for the protection of technical investment for existing applications. This release of the XFS specification has been prompted by a series of factors.

There has been a technical imperative to extend the scope of the existing specification to include new devices, such as the Barcode Reader, Card Dispenser and Item Processing Module.

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

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 XFS 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. 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.
- Enhanced Audio Controller, 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 Enhanced Audio Controller Overview

The Enhanced Audio Controller is provided to support the requirements of the American Disabilities Act. The Enhanced Audio Controller device controls how private and public audio are broadcast when a headset is inserted into/removed from the Audio Jack, and when the Handset is off-hook/on-hook. In the following ‘Privacy Device’ is used to refer to either the headset or handset. This device allows audio feedback publicly and/or via the consumer’s Privacy Device (vendor hardware permitting). For privacy, the device allows input to only be directed to the consumers’ Privacy Device. In ‘auto’ and ‘semi-auto’ mode (and where the vendor’s hardware allows), public transmission of audio can be automatically inhibited when the consumer’s Privacy Device is activated. In ‘auto’ mode (and where the vendor’s hardware allows), public transmission of audio can be automatically re-activated when the consumer’s Privacy Device is deactivated.

The Enhanced Audio Controller provides the application with the following information:

- If a Privacy Device is activated (headset connected/handset off the hook).
- Whether the audio output is to the speakers or to the Privacy Device.
- Privacy/public mode: i.e. whether the activation of the Privacy Device automatically switches public audio on or off.

The device is managed by the sensors WFS_SIU_ENHANCEDAUDIO, WFS_SIU_HANDSETSENSOR, 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_HANDSETSENSOR sensor is used to:

- Provide information on the presence of the handset device.
- To report whether a handset is currently off the hook.
- Report state change events when a handset is taken off the hook or put on the hook.

The WFS_SIU_ENHANCEDAUDIOCONTROL auxiliary is used to control the behavior of the Enhanced Audio Controller. It allows the application to:

- Set the mode of the Enhanced Audio Controller - auto mode, semi-auto mode or manual mode.
- Set the state of the Enhanced Audio Controller- public or private.

There are no events associated with this auxiliary.

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

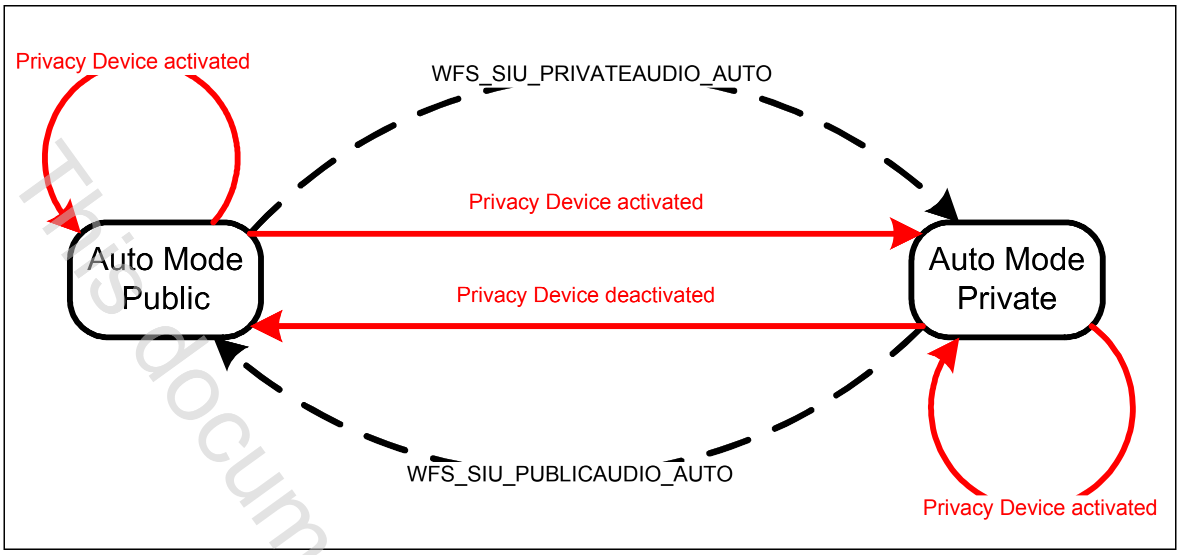
The following describes the device behavior during auto and manual mode.

Auto Mode

In auto mode, when a consumer activates the Privacy Device, the audio is automatically directed to the Privacy Device and the audio is no longer sent to the speakers. When the Privacy Device is deactivated, the audio is redirected to the speakers. The following state diagram completely describes the behavior 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’s Privacy Device 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 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.

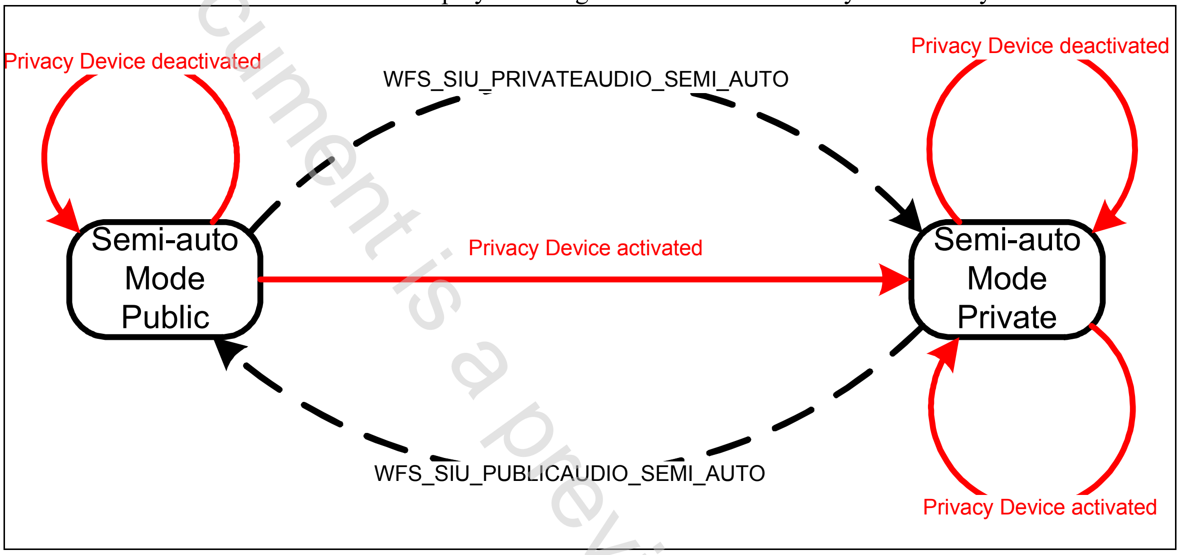
Semi-Auto Mode

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

In semi-auto mode, when a consumer's Privacy Device is activated, the audio is automatically directed to the Privacy Device and the audio is no longer sent to the speakers. When the Privacy Device is deactivated the audio remains directed at the existing interface (i.e. not the speakers). 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 behavior 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's Privacy Device only.


Privacy Device deactivated
WFS_SIU_PRIVATEAUDIO_SEMI_AUTO
Privacy Device deactivated
Semi-auto Mode Public
Privacy Device activated
Semi-auto Mode Private
WFS_SIU_PUBLICAUDIO_SEMI_AUTO
Privacy Device activated

Semi-Auto Mode State Diagram 2

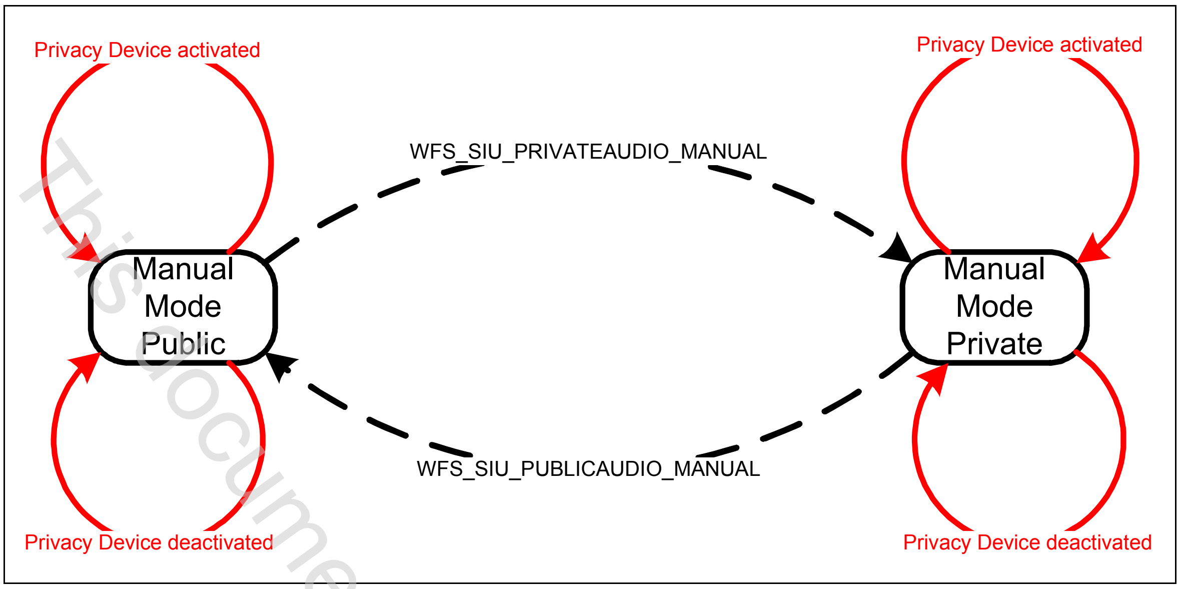
The dashed-line transitions are caused by application calls to WFS_CMD_SIU_SET_PORT or WFS_CMD_SIU_SET_AUXILIARY for the WFS_SIU_ENHANCEDAUDIOCONTROL auxiliary with values of WFS_SIU_PRIVATEAUDIO_AUTO or WFS_SIU_PUBLICAUDIO_AUTO.

Manual Mode

In manual mode, when a consumer's Privacy Device is activated, 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 behavior 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's Privacy Device only.


Privacy Device activated
Privacy Device activated
WFS_SIU_PRIVATEAUDIO_MANUAL
Manual Mode Public
Manual Mode Private
WFS_SIU_PUBLICAUDIO_MANUAL
Privacy Device deactivated
Privacy Device deactivated

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.

Inter-Mode Behavior

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 a connected Privacy Device, as well as the speakers. Independent volume control of the speakers and Privacy Device is not supported.
- Any ‘beep’ tones generated by the PINPAD, etc will be fed to a connected Privacy Device (vendor hardware permitting).

3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer's Reference Revision 3.10