CEN

WORKSHOP

AGREEMENT

CWA 16926-69

January 2023

ICS 35.200; 35.240.15; 35.240.40

English version

Extensions for Financial Services (XFS) interface specification Release 3.50 - Part 69: Sensors and Indicators Unit Device Class Interface - Programmer's Reference - Migration from Version 3.40 (CWA 16926:2020) to Version 3.50 (this CWA)

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-CENELEC Management Centre 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, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Republic of North Macedonia, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Türkiye and United Kingdom.

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

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

Ref. No.:CWA 16926-69:2023 E

Table of Contents

European Foreword

This CEN Workshop Agreement has been developed in accordance with the CEN-CENELEC Guide 29 “CEN/CENELEC Workshop Agreements – The way to rapid consensus” and with the relevant provisions of CEN/CENELEC Internal Regulations – Part 2. It was approved by a Workshop of representatives of interested parties on 2022-11-08, the constitution of which was supported by CEN following several public calls for participation, the first of which was made on 1998-06-24. However, this CEN Workshop Agreement does not necessarily include all relevant stakeholders.

The final text of this CEN Workshop Agreement was provided to CEN for publication on 2022-11-18.

The following organizations and individuals developed and approved this CEN Workshop Agreement:

- AURIGA SPA
- CIMA SPA
- DIEBOLD NIXDORF SYSTEMS GMBH
- FIS BANKING SOLUTIONS UK LTD (OTS)
- FUJITSU TECHNOLOGY SOLUTIONS
- GLORY LTD
- GRG BANKING EQUIPMENT HK CO LTD
- HITACHI CHANNEL SOLUTIONS CORP
- HYOSUNG TNS INC
- JIANGSU GUOGUANG ELECTRONIC INFORMATION TECHNOLOGY
- KAL
- KEBA HANDOVER AUTOMATION GMBH
- NCR FSG
- NEXUS SOFTWARE
- OBERTHUR CASH PROTECTION
- OKI ELECTRIC INDUSTRY SHENZHEN
- SALZBURGER BANKEN SOFTWARE
- SECURE INNOVATION
- SIGMA SPA

It is possible that some elements of this CEN/CWA may be subject to patent rights. The CEN-CENELEC policy on patent rights is set out in CEN-CENELEC Guide 8 “Guidelines for Implementation of the Common IPR Policy on Patents (and other statutory intellectual property rights based on inventions)”. CEN shall not be held responsible for identifying any or all such patent rights.

The Workshop participants have made every effort to ensure the reliability and accuracy of the technical and non-technical content of CWA 16926-10, but this does not guarantee, either explicitly or implicitly, its correctness. Users of CWA 16926-10 should be aware that neither the Workshop participants, nor CEN can be held liable for damages

or losses of any kind whatsoever which may arise from its application. Users of CWA 16926-10 do so on their own responsibility and at their own risk.

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

Part 19: Biometrics Device Class Interface - Programmer's Reference

Parts 20 - 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

Part 48: XFS MIB Device Specific Definitions - Biometrics Device Class

Parts 49 - 60 are reserved for future use.

Part 61: Application Programming Interface (API) - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Service Provider Interface (SPI) - Programmer's Reference

Part 62: Printer and Scanning Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 63: Identification Card Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 65: PIN Keypad Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 67: Depository Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 71: Camera Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 72: Alarm Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 74: Cash-In Module Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 75: Card Dispenser Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 76: Barcode Reader Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 77: Item Processing Module Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 78: Biometric Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (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: https://www.cencenelec.eu/areas-of-work/cen-sectors/digital-society-cen/cwa-download-area/.

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

Revision History:

3.00	October 18, 2000	Initial Release.
3.10	November 29, 2007	For a description of changes from version 3.00 to version 3.10 see the SIU 3.10 Migration document.
3.20	March 2, 2011	For a description of changes from version 3.10 to version 3.20 see the SIU 3.20 Migration document.
3.30	March 19, 2015	For a description of changes from version 3.20 to version 3.30 see the SIU 3.30 Migration document.
3.40	December 06, 2019	For a description of changes from version 3.30 to version 3.40 see the SIU 3.40 Migration document.
3.50	November 18, 2022	For a description of changes from version 3.40 to version 3.50 see the SIU 3.50 Migration document.

1. Introduction

1.1 Background to Release 3.50

The CEN/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 (European Committee for Standardization/Information Society Standardization System) Workshop environment. CEN Workshops aim to arrive at a European consensus on an issue that can be published as a CEN Workshop Agreement (CWA).

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

Release 3.50 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 specification extends the functionality and capabilities of the existing devices covered by the specification:

- Addition of E2E security
- PIN Password Entry

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 for Execute commands or WFS_ERR_UNSUPP_CATEGORY error for Info commands is returned to the calling application. An example would be a request from an application to a cash dispenser to retract items where the dispenser hardware does not have that capability; the Service Provider recognizes the command but, since the cash dispenser it is managing is unable to fulfil the request, 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 for Execute commands or WFS_ERR_INVALID_CATEGORY error for Info commands 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 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.
  Note that while the SIU device class provides some basic support for guidance lights, extended guidance light functionality is specified in the individual device class specifications. Therefore it is recommended that device guidance lights be supported and controlled via the individual device classes.
- 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.

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

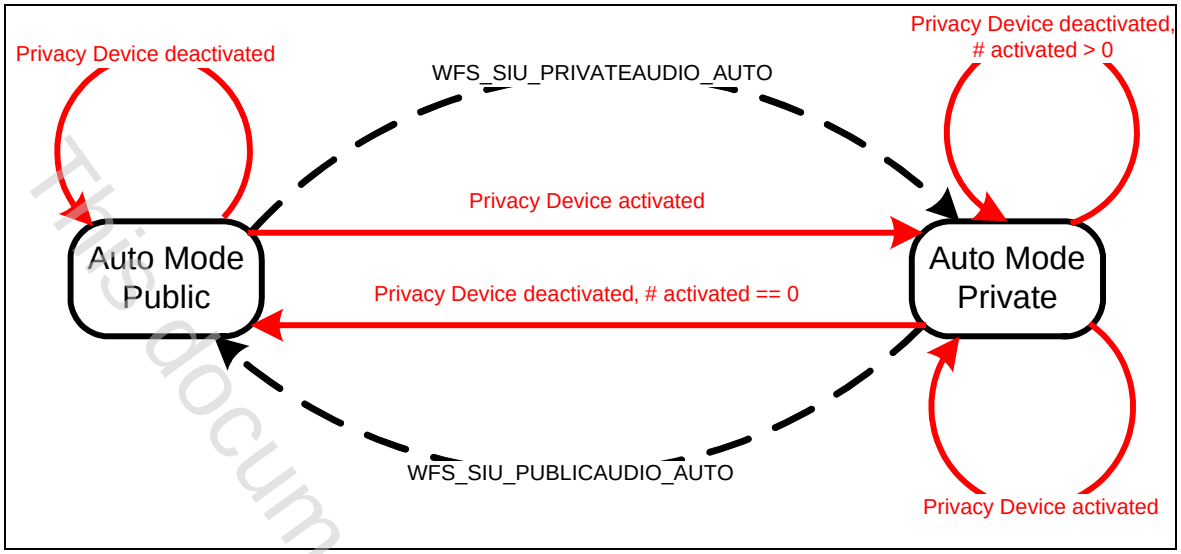
The following describes the device behavior during auto and manual mode.

Auto Mode

In auto mode, when a consumer activates a 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. If more than one Privacy Device has been activated, audio is not redirected to the speakers until all Privacy Devices have been deactivated. 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_PORTS 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.

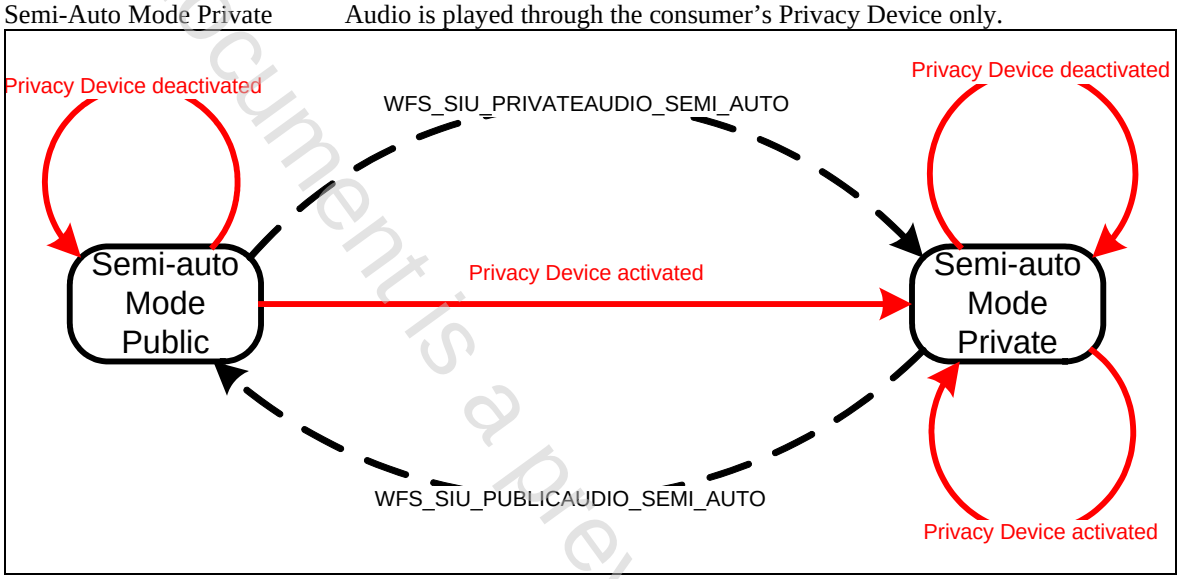
Semi-Auto Mode

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.



Semi-Auto Mode State Diagram 2

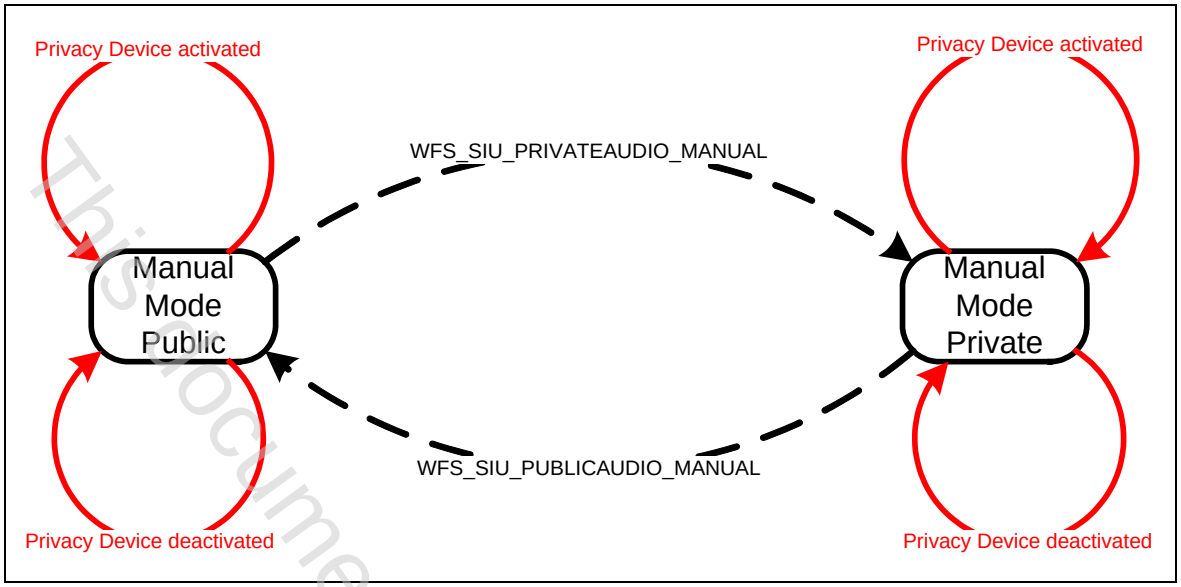
The dashed-line transitions are caused by application calls to WFS_CMD_SIU_SET_PORTS or WFS_CMD_SIU_SET_AUXILIARY for the WFS_SIU_ENHANCEDAUDIOCONTROL auxiliary with values of WFS_SIU_PRIVATEAUDIO_SEMI_AUTO or WFS_SIU_PUBLICAUDIO_SEMI_AUTO.

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.



Manual Mode State Diagram 1

The dashed-line transitions are caused by application calls to WFS_CMD_SIU_SET_PORTS 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_PORTS, etc. with the WFS_SIU_PUBLICAUDIO_AUTO, WFS_SIU_PRIVATEAUDIO_AUTO, WFS_SIU_PUBLICAUDIO_SEMI_AUTO, and WFS_SIU_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).

2.2 Enhanced Microphone Controller Overview

The Enhanced Microphone Controller is provided to support a system of one or more microphones. Its behavior is very similar to that of the Enhanced Audio Controller. The Enhanced Microphone Controller device controls how private and public audio input behave when a headset microphone is inserted into/removed from the Audio/Microphone Jack, and when the Handset with an integrated microphone is off-hook/on-hook. The device allows audio input publicly (via a microphone in the fascia) and/or via the consumer's Privacy Device (vendor hardware permitting). For improved audio clarity or 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 Microphone Controller provides the application with the following information:

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

The device is managed by the sensors WFS_SIU_HEADSETMICROPHONE, WFS_SIU_HANDSETSENSOR, and an auxiliary WFS_SIU_ENHANCEDMICROPHONECONTROL.

The WFS_SIU_HEADSETMICROPHONE sensor is used to:

- Provide information on the presence of the Microphone Jack device.
- To report whether a headset/external microphone is currently attached.
- Report state change events when a headset/external microphone is inserted or removed.

Some systems may contain a headset jack that enables both audio input and output via a single jack. In this case, the WFS_SIU_ENHANCEDAUDIO capability will report WFS_SIU_BIDIRECTIONAL, and when a bi-directional headset is inserted into the jack, the status for both the WFS_SIU_ENHANCEDAUDIO and WFS_SIU_HEADSETMICROPHONE ports will change appropriately.

The WFS_SIU_HANDSETSENSOR sensor is the same sensor used for private audio output. However, the WFS_SIU_HANDSETSENSOR capability flag WFS_SIU_MICROPHONE indicates whether the Handset also contains an integrated microphone for audio input.

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

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

The Enhanced Microphone Controller modes and state transitions are very much like those for the Enhanced Audio Controller. The Enhanced Microphone Controller can be set to auto, semi-auto or manual modes. State transitions occur in the same way that they are described in the state diagrams in the previous section, either when changed manually by the application or when a Privacy Devices is activated/deactivated (depending on the mode). Note that if the WFS_SIU_FASCIAMICROPHONE indicates WFS_SIU_NOT_AVAILABLE, none of the public modes for the microphone will be supported. The Enhanced Audio Controller and Enhanced Microphone Controller states are independent of each other, but can be linked under certain transitions while in auto mode or semi-auto mode. For example, if a handset that contains microphone is activated, both controllers can transition to the private state. On the other hand, if a handset that does not contain a microphone is activated, the Enhanced Audio Controller may transition to the private state while the Enhanced Microphone Controller remains in the public state. Likewise, if a headset that contains a microphone is plugged into a jack that supports integrated audio input and output, both controllers can transition to the private state. Note that in this case, both the WFS_SIU_ENHANCEDAUDIO and WFS_SIU_HEADSETMICROPHONE sensors would change state to WFS_SIU_PRESENT regardless of the mode.

3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer's Reference Revision 3.50
2. PCI Security Standards Council PCI DSS v3.1 Requirements and Security Assessment Procedures https://www.pcisecuritystandards.org/documents/PCI_DSS_v3-1.pdf