SLOVENSKI STANDARD
SIST CWA 16926-63:2023

01-april-2023

Specifikacija vmesnika razširitev za finančne storitve (XFS), izdaja 3.50 - 63. del: Vmesnik razreda naprav identifikacijskih kartic - Referenca za programerje - Prehod z različice 3.40 (CWA 16926:2020) na različico 3.50 (ta CWA)

Extensions for Financial Services (XFS) interface specification Release 3.50 - Part 63: Identification Card Device Class Interface - Programmer's Reference - Migration from Version 3.40 (CWA 16926:2020) to Version 3.50 (this CWA)

Ta slovenski standard je istoveten z: **CWA 16926-63:2023**

ICS:

35.200	Vmesniška in povezovalna oprema	Interface and interconnection equipment
35.240.15	Identifikacijske kartice. Čipne kartice. Biometrija	Identification cards. Chip cards. Biometrics
35.240.40	Uporabniške rešitve IT v bančništvu	IT applications in banking

SIST CWA 16926-63:2023 **en,fr,de**

CEN WORKSHOP AGREEMENT

CWA 16926-63

January 2023

ICS 35.200; 35.240.15; 35.240.40

English version

Extensions for Financial Services (XFS) interface specification Release 3.50 - Part 63: Identification Card 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

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 16926-4, but this does not guarantee, either explicitly or implicitly, its correctness. Users of CWA 16926-4 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 16926-4 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 IDC 3.10 Migration document.
3.20	March 2, 2011	For a description of changes from version 3.10 to version 3.20 see the IDC 3.20 Migration document.
3.30	March 19, 2015	For a description of changes from version 3.20 to version 3.30 see the IDC 3.30 Migration document.
3.40	December 06, 2019	For a description of changes from version 3.30 to version 3.40 see the IDC 3.40 Migration document.
3.50	November 18, 2022	For a description of changes from version 3.40 to version 3.50 see the IDC 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. Identification Card Readers and Writers

This section describes the functions provided by a generic identification card reader/writer service (IDC). These descriptions include definitions of the service-specific commands that can be issued, using the **WFSAsyncExecute**, **WFSExecute**, **WFSGetInfo** and **WFSAsyncGetInfo** functions.

This service allows for the operation of the following categories of units:

- motor driven card reader/writer
- pull through card reader (writing facilities only partially included)
- dip reader
- contactless chip card readers
- permanent chip card readers (each chip is accessed through a unique logical service)

Some motor driven card reader/writers have parking stations inside and can place identification cards there. Once a card is in its parking station another card can be accepted by the card reader. Cards may only be moved out of a parking station if there is no other card present in the media read/write position, the chip I/O position, the transport, or the entry/exit slot.

The following tracks/chips and the corresponding international standards are taken into account in this document:

• Track 1	ISO 7811
• Track 2	ISO 7811
• Track 3	ISO 7811 / ISO 4909
• Cash Transfer Card Track 1	(JIS I: 8 bits/char) Japan
• Cash Transfer Card Track 3	(JIS I: 8 bits/char) Japan
• Front Track 1	(JIS II) Japan
• Watermark	Sweden
• Chip (contacted)	ISO 7816
• Chip (contactless)	ISO 10536, ISO 14443 and ISO 18092

National standards like Transac for France are not considered, but can be easily included via the forms mechanism (see Section 7, Form Definition).

In addition to the pure reading of the tracks mentioned above, security boxes can be used via this service to check the data of writable tracks for manipulation. These boxes (such as CIM or MM) are sensor-equipped devices that are able to check some other information on the card and compare it with the track data.

Persistent values are maintained through power failures, open sessions, close session and system resets.

When the service controls a permanently connected chip card, WFS_ERR_UNSUPP_COMMAND will be returned to all commands except WFS_INF_IDC_STATUS, WFS_INF_IDC_CAPABILITIES, WFS_CMD_IDC_CHIP_POWER, WFS_CMD_IDC_CHIP_IO and WFS_CMD_IDC_RESET.

The following defines the roles and responsibilities of an application within EMV: A distinction needs to be made between EMV Contact support and EMV Contactless support.

When defining an EMV Contact implementation

- EMV Level 2 interaction is handled above the XFS API
- EMV Level 1 interaction is handled below the XFS API

All EMV status information that is defined as a Level 1 responsibility in the EMV specification should be handled below the XFS API.

EMVCo grants EMV Level 1 Approvals to contact IFMs and EMVCo Level 2 Approvals to Application Kernels.

When defining an EMV Contactless implementation

The responsibilities will depend on the type of EMV Contactless Product being implemented.

There are different EMVCo defined product types, they can be found in the EMVCo Type Approval – Contactless Product – Administrative Process document.

- In this specification when referring to the Contactless Product Type – Intelligent Card Reader :

The following must be included and handled below the XFS API:

- An EMVCo Approved Level 1 Contactless PCD
- Entry Point and POS System Architecture according to Book A and B
- EMV Kernels according to Book C1 to C7 (minimum one kernel needs to be supported)

The Network Interface & the Consumer, Merchant Interfaces will be managed above the XFS API.

2.1 Support for EMV Intelligent Contactless Card Readers

In relation to contactless transactions, the terminology used in this document is based on the EMV Contactless Specifications for Payment Systems, see the References section.

There are a number of types of payment systems (or EMV) compliant contactless card readers, from the intelligent reader device; where the reader device handles most of the transaction processing and only returns the result, to a transparent card reader; where the contactless card reader device provides a generic communication channel to the card without having any in-built transaction processing capabilities.

A contactless payment system transaction can be performed in two different ways, magnetic stripe emulation; where the data returned from the chip is formatted as if it was read from the magnetic stripe, and EMV-like; where, in a similar way to a contact EMV transaction, the chip returns a full set of BER-TLV (Basic Encoding Rules-Tag Length Value) data. Each payment system defines when each type, or profile, is used for a transaction, but it is usually dependent on both the configuration of the terminal and contactless card being tapped.

This document will use "magnetic stripe emulation" and "EMV-like" to identify the two profiles of contactless transactions.

Support for a generic contactless communication channel to the card is provided via the WFS_CMD_IDC_CHIP_IO command. This is suitable for use with a transparent contactless card reader or with an intelligent contactless card reader device operating in a pass through mode.

The WFS_CMD_IDC_READ_RAW_DATA command can be used with an intelligent contactless card reader device to provide magnetic track emulation transactions. Only magnetic track emulation transactions can be supported using this command.

When using an intelligent contactless card reader to support both EMV-like and magnetic track emulation transactions a number of commands are required. The WFS_CMD_IDC_EMVCLESS_CONFIGURE command allows the exchange of data to configure the reader for card acceptance and the WFS_CMD_IDC_EMVCLESS_PERFORM_TRANSACTION command enables the reader and performs the transaction with the card when it is tapped. In most cases all the transaction steps involving the card are completed within the initial card tap. Section 9, Appendix provides a sequence diagram showing the expected IDC command sequences, as well as the cardholder and application actions when performing a contactless card based transaction.

Some contactless payment systems allow a 2nd tap of the contactless card. For example a 2nd tap can be used to process authorization data received from the host. In the case of issuer update data this second tap is performed via the WFS_CMD_IDC_EMVCLESS_ISSUERUPDATE command. Section 9, Appendix provides a sequence diagram showing the expected IDC command sequences, as well as the cardholder and application actions. The WFS_INF_IDC_EMVCLESS_QUERY_APPLICATIONS and WFS_CMD_IDC_EMVCLESS_CONFIGURE commands specified later in this document refer to the EMV terminology "Application Identifier (AID) - Kernel Combinations". A detailed explanation can be found in Reference [2] and Reference [3] documents.

This document refers to BER-TLV tags. These are defined by each individual payment systems and contain the data exchanged between the application, contactless card and an intelligent contactless card reader. They are used to configure and prepare the intelligent contactless card reader for a transaction and are also part of the data that is returned by the reader on completion of the cards tap.

Based on the applicable payment system the application is expected to know which tags are required to be configured, what values to use for the tags and how to interpret the tags returned. Intelligent readers are expected to know the BER-TLV tag definitions supported per payment system application. The tags provided in this document are examples of the types of tags applicable to each command. They are not intended to be a definite list.

3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer's Reference Revision 3.~~40~~50.
2. EMVCo Integrated Circuit Card Specifications for Payment Systems Version 4.3
3. EMVCo Contactless Specifications for Payment Systems, Version 2.4
4. EMVCo Contactless Type Approval Administrative Process Version 2.4

4. Info Commands

4.1 WFS_INF_IDC_STATUS

Description This command reports the full range of information available, including the information that is provided either by the Service Provider or, if present, by any of the security modules. In addition to that, the number of cards retained is transmitted for motor driven card reader/writer (for devices of the other categories this number is always set to zero).

Input Param None.

Output Param LPWFSIDCSTATUS lpStatus;

```
typedef struct _wfs_idc_status
     {
     WORD                    fwDevice;
     WORD                    fwMedia;
     WORD                    fwRetainBin;
     WORD                    fwSecurity;
     USHORT                  usCards;
     WORD                    fwChipPower;
     LPSTR                   lpszExtra;
     DWORD                   dwGuidLights[WFS_IDC_GUIDLIGHTS_SIZE];
     WORD                    fwChipModule;
     WORD                    fwMagReadModule;
     WORD                    fwMagWriteModule;
     WORD                    fwFrontImageModule;
     WORD                    fwBackImageModule;
     WORD                    wDevicePosition;
     USHORT                  usPowerSaveRecoveryTime;
     LPWORD                  lpwParkingStationMedia;
     WORD                    wAntiFraudModule;
     } WFSIDCSTATUS, *LPWFSIDCSTATUS;
```

fwDevice
Specifies the state of the ID card device as one of the following flags:

Value	Meaning
WFS_IDC_DEVONLINE	The device is present, powered on and online (i.e. operational, not busy processing a request and not in an error state).
WFS_IDC_DEVOFFLINE	The device is offline (e.g. the operator has taken the device offline by turning a switch).
WFS_IDC_DEVPOWEROFF	The device is powered off or physically not connected.
WFS_IDC_DEVNODEVICE	There is no device intended to be there; e.g. this type of self service machine does not contain such a device or it is internally not configured.
WFS_IDC_DEVHWERROR	The device is present but inoperable due to a hardware fault that prevents it from being used.
WFS_IDC_DEVUSERERROR	The device is present but a person is preventing proper device operation. The application should suspend the device operation or remove the device from service until the Service Provider generates a device state change event indicating the condition of the device has changed e.g. the error is removed (WFS_IDC_DEVONLINE) or a permanent error condition has occurred (WFS_IDC_DEVHWERROR).
WFS_IDC_DEVBUSY	The device is busy and unable to process an Execute command at this time.

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

fwMedia
Specifies the state of the ID card unit as one of the following values. This status is independent of any media in the parking stations.

Value	Meaning
WFS_IDC_MEDIAPRESENT	Media is present in the device, not in the entering position and not jammed. A card in a parking station is not considered to be present. On the latched dip device, this indicates that the card is present in the device and the card is unlatched.
WFS_IDC_MEDIANOTPRESENT	Media is not present in the device and not at the entering position.
WFS_IDC_MEDIAJAMMED	Media is jammed in the device; operator intervention is required.
WFS_IDC_MEDIANOTSUPP	Capability to report media position is not supported by the device (e.g. a typical swipe reader or contactless chip card reader).
WFS_IDC_MEDIAUNKNOWN	The media state cannot be determined with the device in its current state (e.g. the value of *fwDevice* is WFS_IDC_DEVNODEVICE, WFS_IDC_DEVPOWEROFF, WFS_IDC_DEVOFFLINE, or WFS_IDC_DEVHWERROR).
WFS_IDC_MEDIAENTERING	Media is at the entry/exit slot of a motorized device.
WFS_IDC_MEDIALATCHED	Media is present & latched in a latched dip card unit. This means the card can be used for chip card dialog.

fwRetainBin
Specifies the state of the ID card unit retain bin as one of the following values:

Value	Meaning
WFS_IDC_RETAINBINOK	The retain bin of the ID card unit is in a good state.
.WFS_IDC_RETAINNOTSUPP	The ID card unit does not support retain capability.
WFS_IDC_RETAINBINFULL	The retain bin of the ID card unit is full.
WFS_IDC_RETAINBINHIGH	The retain bin of the ID card unit is nearly full.
WFS_IDC_RETAINBINMISSING	The retain bin of the ID card unit is missing.

fwSecurity
Specifies the state of the security unit as one of the following values:

Value	Meaning
WFS_IDC_SECNOTSUPP	No security module is available.
WFS_IDC_SECNOTREADY	The security module is not ready to process cards or is inoperable.
WFS_IDC_SECOPEN	The security module is open and ready to process cards.