

CEN

CWA 16926-14

WORKSHOP

August 2015

AGREEMENT

ICS 35.240.40; 35.240.15; 35.200

English version

**Extensions for Financial Services (XFS) interface specification
Release 3.30 - Part 14: Card Embossing Unit Device Class
Interface - Programmer's Reference**

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, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and United Kingdom.



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

CEN-CENELEC Management Centre: Avenue Marnix 17, B-1000 Brussels

© 2015 CEN All rights of exploitation in any form and by any means reserved worldwide for CEN national Members.

Ref. No.:CWA 16926-14:2015 E

Table of Contents

European foreword.....	4
1. Introduction.....	7
1.1 Background to Release 3.30	7
1.2 XFS Service-Specific Programming.....	7
2. Card Embossing Units.....	9
3. References	10
4. Info Commands.....	11
4.1 WFS_INF_CEU_STATUS	11
4.2 WFS_INF_CEU_CAPABILITIES	15
4.3 WFS_INF_CEU_FORM_LIST	17
4.4 WFS_INF_CEU_MEDIA_LIST	18
4.5 WFS_INF_CEU_QUERY_FORM.....	19
4.6 WFS_INF_CEU_QUERY_MEDIA.....	20
4.7 WFS_INF_CEU_QUERY_FIELD	22
5. Execute Commands.....	24
5.1 WFS_CMD_CEU_EMBOSS_CARD	24
5.2 WFS_CMD_CEU_RESET	26
5.3 WFS_CMD_CEU_POWER_SAVE_CONTROL	27
5.4 WFS_CMD_CEU_EMBOSS_CARD_EX	28
5.5 WFS_CMD_CEU_SUPPLY_REPLENISH.....	31
5.6 WFS_CMD_CEU_SYNCHRONIZE_COMMAND	32
6. Events.....	33
6.1 WFS_SRVE_CEU_INPUTBINTHRESHOLD	33
6.2 WFS_SRVE_CEU_OUTPUTBINTHRESHOLD	34
6.3 WFS_SRVE_CEU_RETAINBINTHRESHOLD.....	35
6.4 WFS_EXEE_CEU_FIELDERROR	36
6.5 WFS_EXEE_CEU_FIELDWARNING	37
6.6 WFS_SRVE_CEU_MEDIAREMOVED	38
6.7 WFS_SRVE_CEU_MEDIADETECTED.....	39
6.8 WFS_EXEE_CEU_EMBOSS_FAILURE.....	40
6.9 WFS_SRVE_CEU_DEVICEPOSITION	41
6.10 WFS_SRVE_CEU_POWER_SAVE_CHANGE.....	42
6.11 WFS_USRE_CEU_TONERTHRESHOLD.....	43
7. Embossing Form, Field and Media Definitions	44
7.1 Definition Syntax	44
7.2 Embossing Form and Media Measurements.....	45

7.3 Embossing Form Definition 46

7.4 Embossing Field Definition..... 47

7.5 Media Definition..... 48

8. C-Header file 49

European foreword

This CWA is revision 3.30 of the XFS interface specification.

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties on March 19th 2015, the constitution of which was supported by CEN following the public call for participation made on 1998-06-24. 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 3.30.

A list of the individuals and organizations which supported the technical consensus represented by the CEN Workshop Agreement is available from the CEN/XFS Secretariat. The CEN XFS Workshop gathered suppliers as well as banks and other financial service companies.

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) - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Service Provider Interface (SPI) - Programmer's Reference

Part 62: Printer and Scanning Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 63: Identification Card Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 65: PIN Keypad Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 67: Depository Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 71: Camera Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 72: Alarm Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 74: Cash-In Module Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 75: Card Dispenser Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 76: Barcode Reader Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (this CWA) - Programmer's Reference

Part 77: Item Processing Module Device Class Interface - Migration from Version 3.20 (CWA 16374) to Version 3.30 (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/work/areas/ict/ebusiness/pages/ws-xfs.aspx>.

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

The formal process followed by the Workshop in the development of the CEN 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 the CEN Workshop Agreement or possible conflict 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.

The final review/endorsement round for this CWA was started on 2015-01-16 and was successfully closed on 2015-03-19. The final text of this CWA was submitted to CEN for publication on 2015-06-19. The specification is continuously reviewed and commented in the CEN 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 3.30.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CEN [and/or CENELEC] shall not be held responsible for identifying any or all such patent rights.

According to the CEN-CENELEC Internal Regulations, the national standards organizations of the following countries are bound to implement this European Standard: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, Former Yugoslav Republic of Macedonia, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland, Turkey and the United Kingdom.

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

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 CEU 3.10 Migration document.
3.20	March 2, 2011	For a description of changes from version 3.10 to version 3.20 see the CEU 3.20 Migration document.
3.30	March 19, 2015	For a description of changes from version 3.20 to version 3.30 see the CEU 3.30 Migration document.

1. Introduction

1.1 Background to Release 3.30

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 3.30 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, but it does not include any new device classes. Notable enhancements include:

- Enhanced reporting of Shutter Jammed Status and a new Shutter Status event for CDM, CIM and IPM.
- Addition of a Synchronize command for all device classes, in order to allow synchronized action where necessary.
- Directional Guidance Light support.
- Addition of a CIM Deplete Command.
- Support for EMV Intelligent Contactless Readers.
- Support in PIN for Encrypting Touch Screen.
- PIN Authentication functionality.
- New PIN Encryption Protocols added for Chinese market.
- PIN TR34 standard supported.

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. Card Embossing Units

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

Embossing card units are generally viewed by XFS as compound devices with the following capabilities and features:

- Embossing or printing of magnetic stripe card/ smart card.
- Reading/encoding magnetic stripe tracks 1, 2, and 3.
- Reading/writing smart card.
- LCD display/ keypad input.

The XFS services supporting the various embossing card unit components are outlined as follows:

- Embossing or printing of magnetic stripe card/ smart card - Card Embossing Unit (CEU) service.
- Reading/encoding magnetic stripe tracks 1, 2, and 3 - ID Card (IDC) service, however when combined encoding/ embossing is performed the CEU service class is used.
- Reading/writing smart cards - ID Card (IDC) service, however when combined writing smart card/ embossing is performed the CEU service class is used.
- LCD display/ keypad input - Text Terminal Unit (TTU) service.

3. References

<p>1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer's Reference Revision 3.30</p>
--

4. Info Commands

4.1 WFS_INF_CEU_STATUS

Description This command reports the full range of information available, including the information that is provided either by the Service Provider or directly from the device.

Input Param None.

Output Param LPWFSCEUSTATUS lpStatus;

```
typedef struct _wfs_ceu_status
{
    WORD                fwDevice;
    WORD                fwMedia;
    WORD                fwRetainBin;
    WORD                fwOutputBin;
    WORD                fwInputBin;
    USHORT              usTotalCards;
    USHORT              usOutputCards;
    USHORT              usRetainCards;
    LPSTR               lpszExtra;
    WORD                wDevicePosition;
    USHORT              usPowerSaveRecoveryTime;
    WORD                wToner;
    WORD                wAntiFraudModule;
} WFSCEUSTATUS, *LPWFSCEUSTATUS;
```

fwDevice

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

Value	Meaning
WFS_CEU_DEVONLINE	The device is present, powered on and online (i.e. operational, not busy processing a request and not in an error state).
WFS_CEU_DEVOFFLINE	The device is offline (e.g. the operator has taken the device offline by turning a switch).
WFS_CEU_DEVPOWEROFF	The device is powered off or physically not connected.
WFS_CEU_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_CEU_DEVHWERROR	The device is present but inoperable due to a hardware fault that prevents it from being used.
WFS_CEU_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_CEU_DEVONLINE) or a permanent error condition has occurred (WFS_CEU_DEVHWERROR).
WFS_CEU_DEVBUSY	The device is busy and unable to process an execute command at this time.
WFS_CEU_DEVFRAUDATTEMPT	The device is present but is inoperable because it has detected a fraud attempt.

WFS_CEU_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 flags:

Value	Meaning
WFS_CEU_MEDIAPRESENT	Media is present in the device, not in the entering position and not jammed.
WFS_CEU_MEDIANOTPRESENT	Media is not present in the device and not at the entering position.
WFS_CEU_MEDIAJAMMED	Media is jammed in the device; operator intervention is required.
WFS_CEU_MEDIANOTSUPP	Capability to report media position is not supported by the device.
WFS_CEU_MEDIAUNKNOWN	The media state cannot be determined with the device in its current state (e.g. the value of <i>fwDevice</i> is WFS_CEU_DEVNODEVICE, WFS_CEU_DEVPOWEROFF, WFS_CEU_DEVOFFLINE, or WFS_CEU_DEVHWERROR).
WFS_CEU_MEDIAENTERING	Media is at the entry/exit slot.
WFS_CEU_MEDIATOPPER	Topper failure.
WFS_CEU_MEDIAINHOPPER	Card is positioned in input bin.
WFS_CEU_MEDIAOUTHOPPER	Card is positioned in output bin.
WFS_CEU_MEDIAMSRE	Encoding failure.
WFS_CEU_MEDIARETAINED	Card is positioned in retain bin.

fwRetainBin

Specifies the state of the CEU retain bin as one of the following flags:

Value	Meaning
WFS_CEU_RETAINBINOK	The retain bin is in a good state.
WFS_CEU_RETAINBINFULL	The retain bin is full.
WFS_CEU_RETAINBINHIGH	The retain bin is nearly full.
WFS_CEU_RETAINBINNOTSUPP	The retain bin state can not be reported.

fwOutputBin

Specifies the state of the Embossing unit output bin as one of the flags:

Value	Meaning
WFS_CEU_OUTPUTBINOK	The output bin is in a good state.
WFS_CEU_OUTPUTBINFULL	The output bin is full.
WFS_CEU_OUTPUTBINHIGH	The output bin is nearly full.
WFS_CEU_OUTPUTNOTSUPP	The output bin state can not be reported.

fwInputBin

Specifies the state of the Embossing unit input bin as one of the flags:

Value	Meaning
WFS_CEU_INPUTBINOK	The input bin is in a good state.
WFS_CEU_INPUTBINEMPTY	The input bin is empty.
WFS_CEU_INPUTBINLOW	The input bin is nearly empty.
WFS_CEU_INPUTNOTSUPP	The input bin state can not be reported.

usTotalCards

The total number of cards, including those in input bin, output bin, and retain bin.

usOutputCards

The total number of output bin cards.

usRetainCards

The total number of retain bin cards.

lpszExtra

Pointer to a list of vendor-specific, or any other extended, information. The information is returned as a series of “*key=value*” strings so that it is easily extensible by Service Providers. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.

wDevicePosition

Specifies the device position. The device position value is independent of the *fwDevice* value, e.g. when the device position is reported as WFS_CEU_DEVICEINPOSITION, *fwDevice* can have any of the values defined above (including WFS_CEU_DEVONLINE or WFS_CEU_DEVOFFLINE). If the device is not in its normal operating position (i.e. WFS_CEU_DEVICEINPOSITION) then media may not be presented through the normal customer interface. This value is one of the following values:

Value	Meaning
WFS_CEU_DEVICEINPOSITION	The device is in its normal operating position, or is fixed in place and cannot be moved.
WFS_CEU_DEVICENOTINPOSITION	The device has been removed from its normal operating position.
WFS_CEU_DEVICEPOSUNKNOWN	Due to a hardware error or other condition, the position of the device cannot be determined.
WFS_CEU_DEVICEPOSNOTSUPP	The physical device does not have the capability of detecting the position.

usPowerSaveRecoveryTime

Specifies the actual number of seconds required by the device to resume its normal operational state from the current power saving mode. This value is zero if either the power saving mode has not been activated or no power save control is supported.

wToner

Specifies the state of the toner or ink supply or the state of the ribbon as one of the following values:

Value	Meaning
WFS_CEU_TONERFULL	The toner or ink supply is full or the ribbon is OK.
WFS_CEU_TONERLOW	The toner or ink supply is low or the print contrast with a ribbon is weak.
WFS_CEU_TONEROUT	The toner or ink supply is empty or the print contrast with a ribbon is not sufficient any more.
WFS_CEU_TONERNOTSUPP	The toner or ink supply is not supported by the device.
WFS_CEU_TONERUNKNOWN	Status of toner or ink supply or the ribbon cannot be determined with device in its current state.

wAntiFraudModule

Specifies the state of the anti-fraud module as one of the following values:

Value	Meaning
WFS_CEU_AFMNOTSUPP	No anti-fraud module is available.
WFS_CEU_AFMOK	Anti-fraud module is in a good state and no foreign device is detected.
WFS_CEU_AFMINOP	Anti-fraud module is inoperable.
WFS_CEU_AFMDEVICEDETECTED	Anti-fraud module detected the presence of a foreign device.
WFS_CEU_AFMUNKNOWN	The state of the anti-fraud module cannot be determined.

Error Codes Only the generic error codes defined in [Ref. 1] can be generated by this command.

Comments Applications which require or expect specific information to be present in the *lpszExtra* parameter

may not be device or vendor-independent.

In the case where communications with the device has been lost, the *fwDevice* field will report WFS_CEU_DEVPOWEROFF when the device has been removed or WFS_CEU_DEVHWERROR if the communications are unexpectedly lost. All other fields should contain a value based on the following rules and priority:

1. Report the value as unknown.
2. Report the value as a general h/w error.
3. Report the value as the last known value.

4.2 WFS_INF_CEU_CAPABILITIES

Description This command is used to retrieve the capabilities of the Card Embossing Unit.

Input Param None.

Output Param LPWFSCEUCAPS lpCaps;

```
typedef struct _wfs_ceu_caps
{
    WORD                wClass;
    BOOL                bCompound;
    BOOL                bCompareMagneticStripe;
    BOOL                bMagneticStripeRead;
    BOOL                bMagneticStripeWrite;
    BOOL                bChipIO;
    WORD                wChipProtocol;
    LPSTR               lpszExtra;
    BOOL                bPowerSaveControl;
    WORD                fwCharSupport;
    WORD                fwType;
    BOOL                bAntiFraudModule;
    LPDWORD              lpdwSynchronizableCommands;
} WFSCEUCAPS, *LPWFSCEUCAPS;
```

wClass

Specifies the logical service class as WFS_SERVICE_CLASS_CEU.

bCompound

Specifies whether the logical device is part of a compound physical device.

bCompareMagneticStripe

Indicates whether CEU has capability of comparing magnetic stripe contents (TRUE) as a prerequisite for an encoding or embossing operation.

bMagneticStripeRead

Indicates whether CEU has magnetic stripe reading capability and is either TRUE or FALSE.

bMagneticStripeWrite

Indicates whether CEU has magnetic stripe writing capability and is either TRUE or FALSE.

bChipIO

Indicates whether CEU has smart card updating capability and is either TRUE or FALSE.

wChipProtocol

Specifies the chip card protocols that are supported by the Service Provider as a combination of the following flags:

Value	Meaning
WFS_CEU_NOTSUPP	The CEU card unit can not handle chip cards.
WFS_CEU_CHIPT0	The CEU card unit can handle the T=0 protocol.
WFS_CEU_CHIPT1	The CEU card unit can handle the T=1 protocol.
WFS_CEU_CHIP_PROTOCOL_NOT_REQUIRED	The CEU card unit is capable of communicating with a chip card without requiring the application to specify any protocol.

lpszExtra

Pointer to a list of vendor-specific, or any other extended, information. The information is returned as a series of “key=value” strings so that it is easily extensible by Service Providers. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.

bPowerSaveControl

Specifies whether power saving control is available. This can either be TRUE if available or FALSE if not available.

fwCharSupport

One or more flags specifying the character sets, in addition to single byte ASCII, that is supported by the Service Provider:

Value	Meaning
WFS_CEU_ASCII	ASCII is supported for XFS forms.
WFS_CEU_UNICODE	UNICODE is supported for XFS forms.

For *fwCharSupport*, a Service Provider can support ONLY ASCII forms or can support BOTH ASCII and UNICODE forms. A Service Provider cannot support UNICODE forms without also supporting ASCII forms.

fwType

Specifies whether the CEU has a card embossing capability and/or a card printing capability. This field will be set to a combination of the following flags:

Value	Meaning
WFS_CEU_EMBOSS	The CEU card unit supports embossing data on cards.
WFS_CEU_PRINT	The CEU card unit supports printing data on cards.

bAntiFraudModule

Specifies whether the anti-fraud module is available. This can either be TRUE if available or FALSE if not available.

lpdwSynchronizableCommands

Pointer to a zero-terminated list of DWORDs which contains the execute command IDs that can be synchronized. If no execute command can be synchronized then this parameter will be NULL.

Error Codes

Only the generic error codes defined in [Ref. 1] can be generated by this command.

Comments

Applications which require or expect specific information to be present in the *lpzExtra* parameter may not be device or vendor-independent.

4.3 WFS_INF_CEU_FORM_LIST

Description	This command is used to retrieve the list of forms available on the device.
Input Param	None.
Output Param	LPSTR lpszFormList; <i>lpszFormList</i> Pointer to a list of null-terminated form names, with the final name terminating with two null characters.
Error Codes	Only the generic error codes defined in [Ref. 1] can be generated by this command.
Comments	None.

4.4 WFS_INF_CEU_MEDIA_LIST

Description	This command is used to retrieve the list of media definitions available on the device.
Input Param	None.
Output Param	LPSTR lpszMediaList; <i>lpszMediaList</i> Pointer to a list of null-terminated media names, with the final name terminating with two null characters.
Error Codes	Only the generic error codes defined in [Ref. 1] can be generated by this command.
Comments	None.

4.5 WFS_INF_CEU_QUERY_FORM

Description This command is used to retrieve details of the definition of a specified CEU form. The WFS_INF_CEU_QUERY_FORM does not currently contain any form attributes, however it is retained for future expansion.

Input Param LPSTR lpzFormName;
lpzFormName
 Points to the null-terminated form name on which to retrieve details.

Output Param LPWFSCEUFORM lpForm;
 typedef struct _wfs_ceu_form
 {
 LPSTR lpzFormName;
 LPSTR lpzFields;
 WORD fwCharSupport;
 WORD wLanguageID;
 } WFSCEUFORM, *LPWFSCEUFORM;

lpzFormName
 Specifies the null-terminated name of the form.

lpzFields
 Pointer to a list of null-terminated field names, with the final name terminating with two null characters.

fwCharSupport
 A single flag specifying the Character Set in which the form is encoded:

Value	Meaning
WFS_CEU_ASCII	ASCII is supported for XFS forms initial data values and FORMAT strings.
WFS_CEU_UNICODE	UNICODE is supported for XFS forms initial data values and FORMAT strings.

wLanguageID
 Specifies the language identifier for the form.

Error Codes In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

Value	Meaning
WFS_ERR_CEU_FORMNOTFOUND	The specified form cannot be found.
WFS_ERR_CEU_FORMINVALID	The specified form is invalid.

Comments None.

4.6 WFS_INF_CEU_QUERY_MEDIA

Description This command is used to retrieve details of the definition of a specified media.

Input Param LPSTR lpszMediaName;

lpszMediaName

Pointer to the null-terminated media name about which to retrieve details.

Output Param LPWFSCEUFRMMEDIA lpFormMedia;

```
typedef struct _wfs_ceu_frm_media
{
    WORD                fwMediaType;
    WORD                wBase;
    WORD                wUnitX;
    WORD                wUnitY;
    WORD                wSizeWidth;
    WORD                wSizeHeight;
    WORD                wEmbossAreaX;
    WORD                wEmbossAreaY;
    WORD                wEmbossAreaWidth;
    WORD                wEmbossAreaHeight;
    WORD                wRestrictedAreaX;
    WORD                wRestrictedAreaY;
    WORD                wRestrictedAreaWidth;
    WORD                wRestrictedAreaHeight;
} WFSCEUFRMMEDIA, *LPWFSCEUFRMMEDIA;
```

fwMediaType

Specifies the type of media as one of the following flags:

Value	Meaning
WFS_CEU_MEDIAECARD	Embossible card media.
WFS_CEU_MEDIAPCARD	Printable card media.

wBase

Specifies the base unit of measurement of the form and can be one of the following:

Value	Meaning
WFS_CEU_INCH	The base unit is inches.
WFS_CEU_MM	The base unit is millimeters.
WFS_CEU_ROWCOLUMN	The base unit is rows and columns.

wUnitX

Specifies the horizontal resolution of the base units as a fraction of the *wBase* value. For example, a value of 16 applied to the base unit WFS_CEU_INCH means that the base horizontal resolution is 1/16".

wUnitY

Specifies the vertical resolution of the base units as a fraction of the *wBase* value. For example, a value of 10 applied to the base unit WFS_CEU_MM means that the base vertical resolution is 0.1 mm.

wSizeWidth

Specifies the width of the media in terms of the base horizontal resolution.

wSizeHeight

Specifies the height of the media in terms of the base vertical resolution.

wEmbossAreaX

Specifies the horizontal offset of the Card Emboss area relative to the top left corner of the media in terms of the base horizontal resolution.

wEmbossAreaY

Specifies the vertical offset of the Card Emboss area relative to the top left corner of the media in terms of the base vertical resolution.

wEmbossAreaWidth

Specifies the Card Emboss area width of the media in terms of the base horizontal resolution.

wEmbossAreaHeight

Specifies the Card Emboss area height of the media in terms of the base vertical resolution.

wRestrictedAreaX

Specifies the horizontal offset of the restricted area relative to the top left corner of the media in terms of the base horizontal resolution.

wRestrictedAreaY

Specifies the vertical offset of the restricted area relative to the top left corner of the media in terms of the base vertical resolution.

wRestrictedAreaWidth

Specifies the restricted area width of the media in terms of the base horizontal resolution.

wRestrictedAreaHeight

Specifies the restricted area height of the media in terms of the base vertical resolution.

Error Codes

In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

Value	Meaning
WFS_ERR_CEU_MEDIANOTFOUND	The specified media definition cannot be found.
WFS_ERR_CEU_MEDIAINVALID	The specified media definition is invalid.

Comments

None.

4.7 WFS_INF_CEU_QUERY_FIELD

Description This function is used to retrieve details on the definition of a single or all fields on a specified form.

Input Param LPWFSCEUQUERYFIELD lpQueryField;

```
typedef struct _wfs_ceu_query_field
{
    LPSTR                lpzFormName;
    LPSTR                lpzFieldName;
} WFSCEUQUERYFIELD, *LPWFSCEUQUERYFIELD;
```

lpzFormName

Points to the null-terminated form name.

lpzFieldName

Points to the null-terminated name of the field about which to retrieve details. If this value is NULL, then retrieve details for all fields on the form. Depending upon whether the form is encoded in UNICODE representation either the *lpzInitialValue* or *lpzUNICODEInitialValue* output fields are used to retrieve the field Initial Value.

Output Param LPWFSCEUFRMFIELD *lppFields;

lppFields

Pointer to a NULL-terminated array of pointers to WFSCEUFRMFIELD structures:

```
typedef struct _wfs_ceu_frm_field
{
    LPSTR                lpzFieldName;
    WORD                fwType;
    WORD                fwClass;
    LPSTR                lpzInitialValue;
    LPSTR                lpzFormat;
    LPWSTR              lpzUNICODEInitialValue;
    LPWSTR              lpzUNICODEFormat;
    WORD                wLanguageID;
} WFSCEUFRMFIELD, *LPWFSCEUFRMFIELD;
```

lpzFieldName

Pointer to the null-terminated field name.

fwType

Specifies the type of field and can be one of the following:

Value	Meaning
WFS_CEU_FIELDTEXT	A text field.
WFS_CEU_FIELDOCR	An Optical Character Recognition (OCR) field.

fwClass

Specifies the class of the field and can be one of the following:

Value	Meaning
WFS_CEU_CLASSSTATIC	The field data cannot be set by the application.
WFS_CEU_CLASSOPTIONAL	The field data can be set by the application.
WFS_CEU_CLASSREQUIRED	The field data must be set by the application.

lpzInitialValue

The initial value of the field when the field is written as output. This value can be NULL if the parameter is not specified in the field definition or the form is encoded in UNICODE.

lpzFormat

Format string as defined in the form for this field. This value can be NULL if the parameter is not specified in the field definition or the form is encoded in UNICODE.

lpzUNICODEInitialValue

The initial value of the field when form is encoded in UNICODE. This value can be NULL if the parameter is not specified in the field definition or the form is not encoded in UNICODE.

lpszUNICODEFormat

Format string as defined in the form for this field when form is encoded in UNICODE. This value can be NULL if the parameter is not specified in the field definition or the form is not encoded in UNICODE.

wLanguageID

Specifies the language identifier for the field.

Error Codes In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

Value	Meaning
WFS_ERR_CEU_FORMNOTFOUND	The specified form cannot be found.
WFS_ERR_CEU_FIELDNOTFOUND	The specified field cannot be found.

Comments None.

5. Execute Commands

5.1 WFS_CMD_CEU_EMBOSS_CARD

Description This command is used to emboss an identification card by merging the supplied variable field data with the defined form and field data specified in the form. Optionally the magnetic stripe can be read and verified before being encoded, or a smart card can be updated.

The ATR of the chip must be obtained before issuing this command by issuing the ID Card class WFS_CMD_IDC_READ_RAW_DATA command.

Input Param LPWFSCEUEMBOSSCARD lpEmbossCard;

```
typedef struct _wfs_ceu_emboss_card
{
    LPSTR          lpszFormName;
    LPSTR          lpzMediaName;
    LPSTR          lpzFields;
    LPSTR          lpzCompareFormIOFormName;
    LPSTR          lpzCompareFormIOTrackData;
    LPSTR          lpzFormIOFormName;
    LPSTR          lpzFormIOTrackData;
    WORD           wChipProtocol;
    ULONG          ulChipDataLength;
    LPBYTE         lpbChipData;
} WFSCEUEMBOSSCARD, *LPWFSCEUEMBOSSCARD;
```

lpzFormName

Pointer to the null-terminated form name.

lpzMediaName

Pointer to the null-terminated media name.

lpzFields

Pointer to a series of "<FieldName>=<Field Value>" strings, where each string is null-terminated with the final string terminating with two null characters.

lpzCompareFormIOFormName

lpzCompareFormIOFormName and *lpzCompareFormIOTrackData* are used collectively when the contents of the magnetic stripe are being read and verified before the card is embossed or the magnetic stripe is encoded. Points to the name of the magnetic stripe form to be used, as defined in the IDC service class.

lpzCompareFormIOTrackData

Points to the data to be used in the form.

lpzFormIOFormName

lpzFormIOFormName and *lpzFormIOTrackData* are used collectively when the magnetic stripe is being encoded (after a successful magnetic stripe compare operation) and during the emboss operation. Points to the name of the form to be used, as defined in the IDC service class.

lpzFormIOTrackData

Points to the data to be used in the form.

wChipProtocol

wChipProtocol, *ulChipDataLength*, and *lpbChipData* are used collectively when the smart card is being updated during the emboss operation. If this parameter equals zero then the smart card should not be updated during the emboss operation. Possible other values are:

Value	Meaning
WFS_CEU_CHIPT0	Use the T=0 protocol to communicate with the chip.
WFS_CEU_CHIPT1	Use the T=1 protocol to communicate with the chip.

WFS_CEU_CHIP_PROTOCOL_NOT_REQUIRED

The Service Provider will automatically determine the protocol used to communicate with the chip.

ulChipDataLength

Specifies the length of the following field *lpbChipData*.

lpbChipData

Points to the data sent to the chip.

Output Param None.

Error Codes In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

Value	Meaning
WFS_ERR_CEU_FORMNOTFOUND	The specified form definition cannot be found.
WFS_ERR_CEU_FORMINVALID	The specified form definition is invalid.
WFS_ERR_CEU_MEDIANOTFOUND	The specified media definition cannot be found.
WFS_ERR_CEU_MEDIAINVALID	The specified media definition is invalid.
WFS_ERR_CEU_NOMEDIA	There is no card inside the device.
WFS_ERR_CEU_MEDIAOVERFLOW	The form overflowed the media.
WFS_ERR_CEU_IDC_FORMNOTFOUND	The specified IDC form definition cannot be found.
WFS_ERR_CEU_IDC_FORMINVALID	The specified IDC form definition is invalid.
WFS_ERR_CEU_INVALIDDATA	An error occurred while communicating with the chip.
WFS_ERR_CEU_PROTOCOLNOTSUPP	The protocol used was not supported by the Service Provider.
WFS_ERR_CEU_ATRNOTOBTAINED	The ATR was not obtained by issuing the IDC class WFS_CMD_CEU_READ_RAW_DATA command.
WFS_ERR_CEU_FIELDSPECFAILURE	The syntax of the <i>lpzFields</i> member is invalid.
WFS_ERR_CEU_FIELDERROR	An error occurred while processing a field, causing termination of the emboss request. An execute event WFS_EXEE_CEU_FIELDERROR is posted with the details.
WFS_ERR_CEU_EMBOSFAILURE	A failure has occurred during Emboss processing. A service event WFS_EXEE_CEU_EMBOSS_FAILURE is posted with details.

Events In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

Value	Meaning
WFS_SRVE_CEU_INPUTBINTHRESHOLD	Input bin is nearly empty.
WFS_SRVE_CEU_OUTPUTBINTHRESHOLD	Output bin is nearly full.
WFS_SRVE_CEU_RETAINBINTHRESHOLD	Retain bin is nearly full.
WFS_EXEE_CEU_EMBOSS_FAILURE	A card embossing failure has occurred.
WFS_EXEE_CEU_FIELDERROR	A fatal error occurred while processing a field.
WFS_EXEE_CEU_FIELDWARNING	A non-fatal error occurred while processing a field.
WFS_SRVE_CEU_MEDIAREMOVED	This event is generated when a card is removed before completion of a write operation.

Comments This command is only supported for backwards compatibility; the WFS_CMD_CEU_EMBOSS_CARD_EX command should instead be used to emboss cards.

5.2 WFS_CMD_CEU_RESET

Description Sends a service reset to the Service Provider. Any media found in the device will be captured into the specified bin (depending on hardware). The WFS_SRVE_CEU_MEDIADETECTED event will indicate that media was found in the device on reset and will indicate the position and status of the media following completion of the command.

Input Param LPWORD lpwCeuMediaControl;

lpwCeuMediaControl

Specifies the action that should be done if media is detected during the reset operation, as one of the following values:

Value	Meaning
WFS_CEU_CTRLTOINPUTBIN	Any media detected should be moved to the input bin.
WFS_CEU_CTRLTOOUTPUTBIN	Any media detected should be moved to the output bin.
WFS_CEU_CTRLTORETAINBIN	Any media detected should be moved to the retain bin.

Output Param None.

Error Codes Only the generic error codes defined in [Ref. 1] can be generated by this command.

Events In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

Value	Meaning
WFS_SRVE_CEU_OUTPUTBINTHRESHOLD	Output bin is nearly full.
WFS_SRVE_CEU_RETAINBINTHRESHOLD	Retain bin is nearly full.
WFS_SRVE_CEU_MEDIADETECTED	Media was detected in the device during a reset.

Comments This command is used by an application control program to cause a device to reset itself to a known good condition.

If *lpwCeuMediaControl* is a NULL pointer the Service Provider will determine where to move any media found.

5.3 WFS_CMD_CEU_POWER_SAVE_CONTROL

Description This command activates or deactivates the power-saving mode.

If the Service Provider receives another execute command while in power saving mode, the Service Provider automatically exits the power saving mode, and executes the requested command. If the Service Provider receives an information command while in power saving mode, the Service Provider will not exit the power saving mode.

Input Param LPWFSCEUPOWERSAVECONTROL lpPowerSaveControl;

```
typedef struct _wfs_ceu_power_save_control
{
    USHORT                usMaxPowerSaveRecoveryTime;
} WFSCEUPOWERSAVECONTROL, *LPWFSCEUPOWERSAVECONTROL;
```

usMaxPowerSaveRecoveryTime

Specifies the maximum number of seconds in which the device must be able to return to its normal operating state when exiting power save mode. The device will be set to the highest possible power save mode within this constraint. If *usMaxPowerSaveRecoveryTime* is set to zero then the device will exit the power saving mode.

Output Param None.

Error Codes In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

Value	Meaning
WFS_ERR_CEU_POWERSAVEVETOOSHORT	The power saving mode has not been activated because the device is not able to resume from the power saving mode within the specified <i>usMaxPowerSaveRecoveryTime</i> value.
WFS_ERR_CEU_POWERSAVEVEMEDIAPRESENT	The power saving mode has not been activated because media is present inside the device.

Events In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

Value	Meaning
WFS_SRVE_CEU_POWER_SAVE_CHANGE	The power save recovery time has changed.

Comments None.

5.4 WFS_CMD_CEU_EMBOSS_CARD_EX

Description This command is used to emboss or print an identification card by merging the supplied variable field data with the defined form and field data specified in the form. Optionally the magnetic stripe can be read and verified before being encoded, or a smart card can be updated.

The ATR of the chip must be obtained before issuing this command by issuing the ID Card class WFS_CMD_IDC_READ_RAW_DATA command.

For backwards compatibility the WFS_CMD_CEU_EMBOSS_CARD command is provided.

Input Param LPWFSCEUEMBOSSCARDEX lpEmbossCardEx;

```
typedef struct _wfs_ceu_emboss_card_ex
{
    LPSTR                lpszFormName;
    LPSTR                lpszMediaName;
    LPSTR                lpszFields;
    LPSTR                lpszCompareFormIOFormName;
    LPSTR                lpszCompareFormIOTrackData;
    LPSTR                lpszFormIOFormName;
    LPSTR                lpszFormIOTrackData;
    WORD                 wChipProtocol;
    ULONG                ulChipDataLength;
    LPBYTE               lpbChipData;
    LPWSTR               lpszUNICODEFields;
} WFSCEUEMBOSSCARDEX, *LPWFSCEUEMBOSSCARDEX;
```

lpszFormName

Pointer to the null-terminated form name.

lpszMediaName

Pointer to the null-terminated media name.

lpszFields

Pointer to a series of "<FieldName>=<Field Value>" strings, where each string is null-terminated with the final string terminating with two null characters. If the field is an index field, then the syntax of the string is instead "<FieldName>[<index>]=<Field Value>", where <index> specifies the zero-based element of the index field.

lpszCompareFormIOFormName

lpszCompareFormIOFormName and *lpszCompareFormIOTrackData* are used collectively when the contents of the magnetic stripe are being read and verified before the card is embossed or the magnetic stripe is encoded. Points to the name of the magnetic stripe form to be used, as defined in the IDC service class.

lpszCompareFormIOTrackData

Points to the data to be used in the form.

lpszFormIOFormName

lpszFormIOFormName and *lpszFormIOTrackData* are used collectively when the magnetic stripe is being encoded (after a successful magnetic stripe compare operation) and during the emboss operation. Points to the name of the form to be used, as defined in the IDC service class.

lpszFormIOTrackData

Points to the data to be used in the form.

wChipProtocol

wChipProtocol, *ulChipDataLength*, and *lpbChipData* are used collectively when the smart card is being updated during the emboss operation. If this parameter equals zero then the smart card should not be updated during the emboss operation. Possible other values are:

Value	Meaning
WFS_CEU_CHIPT0	Use the T=0 protocol to communicate with the chip.
WFS_CEU_CHIPT1	Use the T=1 protocol to communicate with the chip.

WFS_CEU_CHIP_PROTOCOL_NOT_REQUIRED

The Service Provider will automatically determine the protocol used to communicate with the chip.

ulChipDataLength

Specifies the length of the following field *lpbChipData*.

lpbChipData

Points to the data sent to the chip.

lpszUNICODEFields

Pointer to a series of "<FieldName>=<Field Value>" UNICODE strings, where each string is null-terminated with the entire field string terminating with two null characters. If the field is an index field, then the syntax of the string is instead "<FieldName>[<index>]=<Field Value>", where <index> specifies the zero-based element of the index field.

The *lpszUNICODEFields* field should only be used if the form is encoded in UNICODE representation. This can be determined with the WFS_INF_CEU_QUERY_FORM command.

Output Param None.

Error Codes In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

Value	Meaning
WFS_ERR_CEU_FORMNOTFOUND	The specified form definition cannot be found.
WFS_ERR_CEU_FORMINVALID	The specified form definition is invalid.
WFS_ERR_CEU_MEDIANOTFOUND	The specified media definition cannot be found.
WFS_ERR_CEU_MEDIAINVALID	The specified media definition is invalid.
WFS_ERR_CEU_NOMEDIA	There is no card inside the device.
WFS_ERR_CEU_MEDIAOVERFLOW	The form overflowed the media.
WFS_ERR_CEU_IDC_FORMNOTFOUND	The specified IDC form definition cannot be found.
WFS_ERR_CEU_IDC_FORMINVALID	The specified IDC form definition is invalid.
WFS_ERR_CEU_INVALIDDATA	An error occurred while communicating with the chip.
WFS_ERR_CEU_PROTOCOLNOTSUPP	The protocol used was not supported by the service provider.
WFS_ERR_CEU_ATRNOTOBTAINED	The ATR was not obtained by issuing the IDC class WFS_CMD_IDC_READ_RAW_DATA command.
WFS_ERR_CEU_FIELDSPECFAILURE	The syntax of the <i>lpszFields</i> member is invalid.
WFS_ERR_CEU_FIELDERROR	An error occurred while processing a field, causing termination of the emboss request. An execute event WFS_EXEE_CEU_FIELDERROR is posted with the details.
WFS_ERR_CEU_EMBOSFAILURE	A failure has occurred during Emboss or Print processing. A service event WFS_EXEE_CEU_EMBOSS_FAILURE is posted with details.
WFS_ERR_CEU_CHARSETDATA	The character set(s) supported by the Service Provider is inconsistent with the use of the <i>lpszFields</i> or <i>lpszUNICODEFields</i> fields.

Events In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:

Value	Meaning
WFS_SRVE_CEU_INPUTBINTHRESHOLD	Input bin is nearly empty.
WFS_SRVE_CEU_OUTPUTBINTHRESHOLD	Output bin is nearly full.

WFS_SRVE_CEU_RETAINBINTHRESHOLD	Retain bin is nearly full.
WFS_EXEE_CEU_EMOSS_FAILURE	A card embossing or printing failure has occurred.
WFS_EXEE_CEU_FIELDERROR	A fatal error occurred while processing a field.
WFS_EXEE_CEU_FIELDWARNING	A non-fatal error occurred while processing a field.
WFS_SRVE_CEU_MEDIAREMOVED	This event is generated when a card is removed before completion of a write operation.
WFS_USRE_CEU_TONERTHRESHOLD	The toner or ink supply is low or empty or the printing contrast with ribbon is weak or not sufficient, operator intervention is required. Note that this event is sent only once, at the point at which the supply becomes low or empty. It is sent with WFS_CEU_TONERLOW or WFS_CEU_TONEROUT status.

Comments The application will use *lpzFields* or *lpzUNICODEFields* as an input parameter, depending upon the Service Provider capabilities. Legacy (non-UNICODE aware) applications will only use the *lpzFields* field. UNICODE applications can use either the *lpzFields* or *lpzUNICODEFields* fields, provided the Service Provider is UNICODE compliant.

5.5 WFS_CMD_CEU_SUPPLY_REPLENISH

Description	<p>After the supplies have been replenished, this command is used to indicate that one or more supplies have been replenished and are expected to be in a healthy state.</p> <p>Hardware that cannot detect the level of a supply and reports on the supply's status using metrics (or some other means), must assume the supply has been fully replenished after this command is issued. The appropriate threshold event must be broadcast.</p> <p>Hardware that can detect the level of a supply must update its status based on its sensors, generate a threshold event if appropriate, and succeed the command even if the supply has not been replenished. If it has already detected the level and reported the threshold before this command was issued, the command must succeed and no threshold event is required.</p>						
Input Param	<p>LPWFSCBUSUPPLYREPLEN lpSupplyReplen;</p> <pre>typedef struct _wfs_ceu_supply_replen { WORD fwSupplyReplen; } WFSCEBUSUPPLYREPLEN, *LPWFSCBUSUPPLYREPLEN;</pre> <p><i>fwSupplyReplen</i></p> <p>Specifies the supply that was replenished as a combination of the following flags:</p> <table> <tr> <th>Value</th><th>Meaning</th></tr> <tr> <td>WFS_CEU_REPLEN_TONER</td><td>The toner supply was replenished.</td></tr> <tr> <td>WFS_CEU_REPLEN_INPUTBIN</td><td>The input bin supply was replenished.</td></tr> </table>	Value	Meaning	WFS_CEU_REPLEN_TONER	The toner supply was replenished.	WFS_CEU_REPLEN_INPUTBIN	The input bin supply was replenished.
Value	Meaning						
WFS_CEU_REPLEN_TONER	The toner supply was replenished.						
WFS_CEU_REPLEN_INPUTBIN	The input bin supply was replenished.						
Output Param	None.						
Error Codes	Only the generic error codes defined in [Ref. 1] can be generated by this command.						
Events	<p>In addition to the generic events defined in [Ref. 1], the following events can be generated by this command:</p> <table> <tr> <th>Value</th><th>Meaning</th></tr> <tr> <td>WFS_SRVE_CEU_INPUTBINTHRESHOLD</td><td>This service event is used to specify that the state of the input bin supply threshold has been cleared.</td></tr> <tr> <td>WFS_USRE_CEU_TONERTHRESHOLD</td><td>This user event is used to specify that the state of the toner (or ink) supply threshold has been cleared.</td></tr> </table>	Value	Meaning	WFS_SRVE_CEU_INPUTBINTHRESHOLD	This service event is used to specify that the state of the input bin supply threshold has been cleared.	WFS_USRE_CEU_TONERTHRESHOLD	This user event is used to specify that the state of the toner (or ink) supply threshold has been cleared.
Value	Meaning						
WFS_SRVE_CEU_INPUTBINTHRESHOLD	This service event is used to specify that the state of the input bin supply threshold has been cleared.						
WFS_USRE_CEU_TONERTHRESHOLD	This user event is used to specify that the state of the toner (or ink) supply threshold has been cleared.						
Comments	<p>If any one of the specified supplies is not supported by a Service Provider, WFS_ERR_UNSUPP_DATA should be returned, and no replenishment actions will be taken by the Service Provider.</p>						

5.6 WFS_CMD_CEU_SYNCHRONIZE_COMMAND

Description This command is used to reduce response time of a command (e.g. for synchronization with display) as well as to synchronize actions of the different device classes. This command is intended to be used only on hardware which is capable of synchronizing functionality within a single device class or with other device classes.

The list of execute commands which this command supports for synchronization is retrieved in the *lpdwSynchronizableCommands* parameter of the WFS_INF_CEU_CAPABILITIES.

This command is optional, i.e. any other command can be called without having to call it in advance. Any preparation that occurs by calling this command will not affect any other subsequent command. However, any subsequent execute command other than the one that was specified in the *dwCommand* input parameter will execute normally and may invalidate the pending synchronization. In this case the application should call the WFS_CMD_CEU_SYNCHRONIZE_COMMAND again in order to start a synchronization.

Input Param LPWFSCEUSYNCHRONIZECOMMAND *lpSynchronizeCommand*;

```
typedef struct _wfs_ceu_synchronize_command
{
    DWORD dwCommand;
    LPVOID lpCmdData;
} WFSCEUSYNCHRONIZECOMMAND, *LPWFSCEUSYNCHRONIZECOMMAND;
```

dwCommand

The command ID of the command to be synchronized and executed next.

lpCmdData

Pointer to data or a data structure that represents the parameter that is normally associated with the command that is specified in *dwCommand*. For example, if *dwCommand* is WFS_CMD_CEU_RESET then *lpCmdData* will point to a WORD. This parameter can be NULL if no command input parameter is needed or if this detail is not needed to synchronize for the command.

It will be device-dependent whether the synchronization is effective or not in the case where the application synchronizes for a command with this command specifying a parameter but subsequently executes the synchronized command with a different parameter. This case should not result in an error; however, the preparation effect could be different from what the application expects. The application should, therefore, make sure to use the same parameter between *lpCmdData* of this command and the subsequent corresponding execute command.

Output Param None.

Error Codes In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

Value	Meaning
WFS_ERR_CEU_COMMANDUNSUPP	The command specified in the <i>dwCommand</i> field is not supported by the Service Provider.
WFS_ERR_CEU_SYNCHRONIZEUNSUPP	The preparation for the command specified in the <i>dwCommand</i> with the parameter specified in the <i>lpCmdData</i> is not supported by the Service Provider.

Events Only the generic events defined in [Ref. 1] can be generated by this command.

Comments For sample flows of this synchronization see the [Ref 1] Appendix C.

6. Events

6.1 WFS_SRVE_CEU_INPUTBINTHRESHOLD

Description This event specifies that the status of the input bin has changed.

Event Param LPWORD lpwInputBin;

lpwInputBin

Specifies the state of the input bin as one of the following flags:

Value	Meaning
WFS_CEU_INPUTBINOK	The input bin is in a good state.
WFS_CEU_INPUTBINLOW	The input bin is nearly empty.
WFS_CEU_INPUTBINEMPTY	The input bin is empty.

Comments None.

6.2 WFS_SRVE_CEU_OUTPUTBINTHRESHOLD

Description This event specifies that the status of the output bin has changed.

Event Param LPWORD lpwOutputBin;

lpwOutputBin

Specifies the state of the output bin as one of the following flags:

Value	Meaning
WFS_CEU_OUTPUTBINOK	The output bin is in a good state.
WFS_CEU_OUTPUTBINFULL	The output bin is full.
WFS_CEU_OUTPUTBINHIGH	The output bin is nearly full.

Comments None.

6.3 WFS_SRVE_CEU_RETAINBINTHRESHOLD

Description This event specifies that the status of the retain bin has changed.

Event Param LPWORD *lpwRetainBin*;

lpwRetainBin

Specifies the state of the retain bin as one of the following flags:

Value	Meaning
WFS_CEU_RETAINBINOK	The retain bin is in a good state.
WFS_CEU_RETAINBINFULL	The retain bin is full.
WFS_CEU_RETAINBINHIGH	The retain bin is nearly full.

Comments None.

6.4 WFS_EXEE_CEU_FIELDERROR

Description This event specifies that a fatal error has occurred while processing a field.

Event Param LPWFSCEUFIELDFAIL lpFieldFail;

```
typedef struct _wfs_ceu_field_failure
{
    LPSTR          lpzFormName;
    LPSTR          lpzFieldName;
    WORD          wFailure;
} WFSCEUFIELDFAIL, *LPWFSCEUFIELDFAIL;
```

lpzFormName

Points to the null-terminated form name.

lpzFieldName

Points to the null-terminated field name.

wFailure

Specifies the type of failure and can be one of the following:

Value	Meaning
WFS_CEU_FIELDREQUIRED	The specified field <i>must</i> be supplied by the application.
WFS_CEU_FIELDSTATICOVWR	The specified field is static and thus <i>cannot</i> be overwritten by the application.
WFS_CEU_FIELDOVERFLOW	The value supplied for the specified fields is too long.
WFS_CEU_FIELDNOTFOUND	The specified field does not exist.
WFS_CEU_FIELDNOTREAD	The specified field is not an input field.
WFS_CEU_FIELDNOTWRITE	An attempt was made to write to an input field.
WFS_CEU_FIELDHWERROR	The specified field uses special hardware (e.g. OCR) and an error occurred.
WFS_CEU_FIELDTYPENOTSUPPORTED	The form field type is not supported with device.
WFS_CEU_CHARSETFORM	The Service Provider does not support the character set specified in the form.

Comments None.

6.5 WFS_EXEE_CEU_FIELDWARNING

Description	This event is used to specify that a non-fatal error has occurred while processing a field.
Event Param	LPWFSCEUFIELDFAIL lpFieldFail; As defined in the section describing WFS_EXEE_CEU_FIELDERROR.
Comments	None.

6.6 WFS_SRVE_CEU_MEDIAREMOVED

Description	This event is generated when a card is removed before completion of a write operation.
Event Param	None.
Comments	None.

6.7 WFS_SRVE_CEU_MEDIADETECTED

Description This event is generated when a media is detected in the device during a reset operation.

Event Param LPWORD lpwPosition;

lpwPosition

Specifies the media position after the reset operation, as one of the following values:

Value	Meaning
WFS_CEU_MEDIARETAINED	The media was successfully retained during the reset operation.
WFS_CEU_MEDIAREMOVED	The media was removed during the reset operation.
WFS_CEU_MEDIAJAMMED	The media is jammed in the device.
WFS_CEU_MEDIAUNKNOWN	The media is in an unknown position.

Comments None.

6.8 WFS_EXEE_CEU_EMBOSS_FAILURE

Description This service event is used to specify that an error has occurred during processing of a WFS_CMD_CEU_EMBOSS_CARD or WFS_CMD_CEU_EMBOSS_CARD_EX execute command.

Event Param LPWORD lpwEmbossFailure;
lpwEmbossFailure
 Specified as one of the following flags:

Value	Meaning
WFS_CEU_STEPPER_ERROR	Stepper hardware error.
WFS_CEU_TOPPER_FOIL_BREAK	Topper foil has broken.
WFS_CEU_CARD_FEED_ERROR	Card feed failure.
WFS_CEU_MAGNETIC_STRIPE_ERROR	Magnetic stripe read/write error.
WFS_CEU_RETAIN_BIN_FULL	Retain bin is full.
WFS_CEU_OUTPUT_BIN_FULL	Output bin is full.
WFS_CEU_COVER_OPEN	Device cover is open.
WFS_CEU_TOPPER_JAM	Topper has jammed.
WFS_CEU_STACKER_ERROR	Stacker error either inside device or in output bin.
WFS_CEU_SYSTEM_ERROR	Unknown system error.
WFS_CEU_OCR_ERROR	OCR unit failure.
WFS_CEU_EMBOSS_LIMITS_EXCEEDED	Embossing limits exceeded.
WFS_CEU_COMMUNICATIONS_FAILURE	Communications failure.
WFS_CEU_DATA_FORMAT_ERROR	Communications data format error.
WFS_CEU_BUFFER_OVERRUN	Buffer overrun.
WFS_CEU_PRE_ENCODE_READ_ERROR	Pre-encode read error.
WFS_CEU_PRE_ENCODE_DATA_MATCH_ERROR	Data has failed to compare during pre-encode data match step.
WFS_CEU_INPUT_BIN_EMPTY	Input bin is empty.
WFS_CEU_DEVICE_BUSY	Device is busy, unable to emboss card.
WFS_CEU_TONER_OUT_ERROR	Toner or ink supply is empty or printing contrast with ribbon is not sufficient.
WFS_CEU_MEDIA_JAM	The card is jammed. Operator intervention is required.

Comments None.

6.9 WFS_SRVE_CEU_DEVICEPOSITION

Description This service event reports that the device has changed its position status.

Event Param LPWFSCEUDEVICEPOSITION lpDevicePosition;

```
typedef struct _wfs_ceu_device_position
{
    WORD wPosition;
} WFSCEUDEVICEPOSITION, *LPWFSCEUDEVICEPOSITION;
```

wPosition

Position of the device as one of the following values:

Value	Meaning
WFS_CEU_DEVICEINPOSITION	The device is in its normal operating position.
WFS_CEU_DEVICENOTINPOSITION	The device has been removed from its normal operating position.
WFS_CEU_DEVICEPOSUNKNOWN	The position of the device cannot be determined.

Comments None.

6.10 WFS_SRVE_CEU_POWER_SAVE_CHANGE

Description	This service event specifies that the power save recovery time has changed.
Event Param	<p>LPWFSCEUPOWERSAVECHANGE lpPowerSaveChange;</p> <pre>typedef struct _wfs_ceu_power_save_change { USHORT usPowerSaveRecoveryTime; } WFSCEUPOWERSAVECHANGE, *LPWFSCEUPOWERSAVECHANGE;</pre> <p><i>usPowerSaveRecoveryTime</i> Specifies the actual number of seconds required by the device to resume its normal operational state. This value is zero if the device exited the power saving mode.</p>
Comments	If another device class compound with this device enters into a power saving mode this device will automatically enter into the same power saving mode and this event will be generated.

6.11 WFS_USRE_CEU_TONERTHRESHOLD

Description This user event is used to specify that the state of the toner or ink supply or the state of the ribbon reached a threshold.

Event Param LPWFSCEUTONERSTATUS lpTonerStatus;

```
typedef struct _wfs_ceu_toner_status
{
    LPWORD lpwTonerThreshold;
} WFSCEUTONERSTATUS, *LPWFSCEUTONERSTATUS;
```

lpwTonerThreshold

Specifies the current state of the toner or ink supply or the state of the ribbon as one of the following values:

Value	Meaning
WFS_CEU_TONERFULL	The toner, ink or ribbon in the printer is in a good state.
WFS_CEU_TONERLOW	The toner or ink in the printer is low or the print contrast with a ribbon is weak.
WFS_CEU_TONEROUT	The toner or ink in the printer is out or the print contrast with a ribbon is not sufficient any more.

Comments None.

7. Embossing Form, Field and Media Definitions

This section outlines the format of the embossing definitions of forms and the fields within them.

7.1 Definition Syntax

The syntactic rules for form, field and media definitions are as follows:

- White space space, tab
- Line continuation backslash (\)
- Line termination CR, LF, CR/LF; line termination ends a “keyword section” (a keyword and its value[s])
- Keywords must be all upper case
- Names (field/media/font names) any case; case is preserved; Service Providers are case sensitive
- Strings all strings must be enclosed in double quote characters ("); to include a double quote in a string, “escape” with a forward slash (/)
- Comments start with two forward slashes (//), end at line termination

Other notes:

- If a keyword is present, all its values must be specified; default values are used only if the keyword is absent.
- Values that are character strings are marked with asterisks in the definitions below, and must be quoted as specified above.
- All forms can be represented using either ISO 646 (ANSI) or UNICODE character encoding. If the UNICODE representation is used then all Names and Strings are restricted to an internal representation of ISO 646 (ANSI) characters. Only the INITIALVALUE and FORMAT keyword values can have double byte values outside of the ISO 646 (ANSI) character set.
- If forms character encoding is UNICODE then, consistent with the UNICODE standard, the file prefix must be in little endian (xFFFE) or big endian (xFEFF) notation, such that UNICODE encoding is recognized.

7.2 Embossing Form and Media Measurements

The UNIT keyword sections of the form and media definitions specify the base horizontal and vertical resolution as follows:

- The *base* value specifies the base unit of measurement.
- The x and y values specify the horizontal and vertical resolution as fractions of the base value (e.g. an x value of 10 and a base value of MM means that the base horizontal resolution is 0.1mm).

The base resolutions thus defined by the UNIT keyword section of the *form* definition are used as the units of the form definition keyword sections:

- SIZE (*width* and *height* values)
- ALIGNMENT (*xoffset* and *yoffset* values)

and of the field definition keyword sections:

- POSITION (*x* and *y* values)
- SIZE (*width* and *height* values)

The base resolutions thus defined by the UNIT keyword section of the *media* definition are used as the units of the media definition keyword sections:

- SIZE (*width* and *height* values)
- EMBOSSAREA (*x*, *y*, *width* and *height* values)
- RESTRICTED (*x*, *y*, *width* and *height* values)

7.3 Embossing Form Definition

XFSFORM		<i>formname</i>	
BEGIN			
(required)	UNIT	<i>base,</i> <i>x,</i> <i>y</i>	Base resolution unit for form definition MM INCH ROWCOLUMN Horizontal base unit fraction Vertical base unit fraction
(required)	SIZE	<i>width,</i> <i>height</i>	Width of form Height of form
	ALIGNMENT	<i>alignment,</i> <i>xoffset,</i> <i>yoffset</i>	Alignment of the form on the physical media: TOPLEFT (default) TOPRIGHT BOTTOMLEFT BOTTOMRIGHT Horizontal offset relative to the horizontal alignment specified by alignment. Always specified as a positive value (i.e. if aligned to the right side of the media, means offset the form to the left). (default = 0) Vertical offset relative to the vertical alignment specified by alignment. Always specified as a positive value (i.e. if aligned to the bottom of the media, means offset the form upward). (default = 0)
	VERSION	<i>major,</i> <i>minor,</i> <i>date*,</i> <i>author*</i>	Major version number Minor version number Creation/modification date Author of form
(required)	LANGUAGE	<i>languageID</i>	Language used in this form - a 16 bit value (LANGID) which is a combination of a primary (10 bits) and a secondary (6 bits) language ID (This is the standard language ID in the Win32 API; standard macros support construction and decomposition of this composite ID)
	COPYRIGHT	<i>copyright*</i>	Copyright entry
	TITLE	<i>title*</i>	Title of form
	COMMENT	<i>comment*</i>	Comment section
	USERPROMPT	<i>prompt*</i>	Prompt string for user interaction
	[XFSFIELD	<i>fieldname</i>	One field definition (as defined in the next section) for each field in the form
	BEGIN ... END]		
END			

7.4 Embossing Field Definition

XFSFIELD		<i>fieldname</i>	
BEGIN			
(required)	POSITION	<i>x,</i> <i>y</i>	Horizontal position (relative to left or right side of form, depending upon HPOSITION keyword) Vertical position (relative to top or bottom of form, depending upon VPOSITION keyword)
	HPOSITION		Horizontal field positioning relative to: LEFT (default) RIGHT
	VPOSITION		Vertical field positioning relative to: TOP BOTTOM (default)
	SIDE		Side of card: FRONT (default) BACK
(required)	SIZE	<i>width,</i> <i>height</i>	Field width Field height
	TYPE	<i>fieldtype</i>	Type of field: TEXT (default) OCR
	CLASS	<i>class</i>	Field class OPTIONAL (default) STATIC REQUIRED
	CASE	<i>case</i>	Convert field contents to NOCHANGE (default) UPPER LOWER
	HORIZONTAL	<i>justify</i>	Horizontal alignment of field contents LEFT (default) RIGHT CENTER JUSTIFY
	VERTICAL	<i>justify</i>	Vertical alignment of field contents BOTTOM (default) CENTER TOP
	FONT	<i>fontname*</i>	Font name; in some cases this predefines the following parameters:
	POINTSIZ	<i>pointsize</i>	Point size
	CPI	<i>cpi</i>	Characters per inch
	LPI	<i>lpi</i>	Lines per inch
	FORMAT	<i>formatstring*</i>	This is an application defined input field describing how the application should format the data. This may be interpreted by the Service Provider.
	INITIALVALUE	<i>value*</i>	Initial value
	LANGUAGE	<i>languageID</i>	Language used in this field – a 16 bit value (LANGID) which is a combination of a primary (10 bits) and a secondary (6 bits) language ID (This is the standard language ID in the Win32 API; standard macros support construction and decomposition of this composite ID) If unspecified defaults to form definition LANGUAGE specification.
END			

7.5 Media Definition

The media definition determines those characteristics that result from the combination of a particular media type together with a particular vendor's identification card or smart card. The aim is to make it easy to move forms between different vendor's identification cards or smart cards which might have different constraints on how they handle a specific media type. It is the Service Provider's responsibility to ensure that the form definition does not specify the embossing of any fields that conflict with the media definition. An example of such a conflict might be that the form definition asks for a field to be embossed in an area that the media definition defines as a restricted area, such as on the chip of a smart card.

XFSMEDIA		<i>medianame*</i>	
BEGIN			
	TYPE	<i>type</i>	Predefined media types are: EMBOSSCARD PRINTCARD
(required)	UNIT	<i>base,</i> <i>x,</i> <i>y</i>	Base resolution unit for media definition MM INCH ROWCOLUMN Horizontal base unit fraction Vertical base unit fraction
(required)	SIZE	<i>width,</i> <i>height</i>	Width of physical media Height of physical media
	EMBOSSAREA	<i>x,</i> <i>y,</i> <i>width,</i> <i>height</i>	Embossing or Printing area relative to top left corner of physical media (default = physical size of media)
	RESTRICTED	<i>x,</i> <i>y,</i> <i>width,</i> <i>height</i>	Restricted area relative to top left corner of physical media (default = no restricted area)
END			

8. C-Header file

```

/*****
 *
 * xfsceu.h      XFS - Card Embossing Unit (CEU) definitions
 *
 *              Version 3.30   (March 19 2015)
 *
 *****/

#ifndef __INC_XFSCEU__H
#define __INC_XFSCEU__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

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

/* values of WFSCEUCAPS.wClass */

#define WFS_SERVICE_CLASS_CEU          (12)
#define WFS_SERVICE_CLASS_NAME_CEU    "CEU"
#define WFS_SERVICE_CLASS_VERSION_CEU (0x1E03) /* Version 3.30 */

#define CEU_SERVICE_OFFSET             (WFS_SERVICE_CLASS_CEU * 100)

/* CEU Info Commands */

#define WFS_INF_CEU_STATUS              (CEU_SERVICE_OFFSET + 1)
#define WFS_INF_CEU_CAPABILITIES        (CEU_SERVICE_OFFSET + 2)
#define WFS_INF_CEU_FORM_LIST           (CEU_SERVICE_OFFSET + 3)
#define WFS_INF_CEU_QUERY_FORM          (CEU_SERVICE_OFFSET + 4)
#define WFS_INF_CEU_MEDIA_LIST          (CEU_SERVICE_OFFSET + 5)
#define WFS_INF_CEU_QUERY_MEDIA         (CEU_SERVICE_OFFSET + 6)
#define WFS_INF_CEU_QUERY_FIELD         (CEU_SERVICE_OFFSET + 7)

/* CEU Execute Commands */

#define WFS_CMD_CEU_EMBOSSE_CARD        (CEU_SERVICE_OFFSET + 1)
#define WFS_CMD_CEU_RESET               (CEU_SERVICE_OFFSET + 2)
#define WFS_CMD_CEU_POWER_SAVE_CONTROL (CEU_SERVICE_OFFSET + 3)
#define WFS_CMD_CEU_EMBOSSE_CARD_EX    (CEU_SERVICE_OFFSET + 4)
#define WFS_CMD_CEU_SUPPLY_REPLENISH    (CEU_SERVICE_OFFSET + 5)
#define WFS_CMD_CEU_SYNCHRONIZE_COMMAND (CEU_SERVICE_OFFSET + 6)

/* CEU Messages */

#define WFS_SRVE_CEU_INPUTBINTHRESHOLD  (CEU_SERVICE_OFFSET + 1)
#define WFS_SRVE_CEU_OUTPUTBINTHRESHOLD (CEU_SERVICE_OFFSET + 2)
#define WFS_SRVE_CEU_RETAINBINTHRESHOLD (CEU_SERVICE_OFFSET + 3)
#define WFS_EXEE_CEU_FIELDERROR         (CEU_SERVICE_OFFSET + 4)
#define WFS_EXEE_CEU_FIELDWARNING       (CEU_SERVICE_OFFSET + 5)
#define WFS_EXEE_CEU_EMBOSSE_FAILURE    (CEU_SERVICE_OFFSET + 6)
#define WFS_SRVE_CEU_MEDIAREMOVED        (CEU_SERVICE_OFFSET + 7)
#define WFS_SRVE_CEU_MEDIADETECTED      (CEU_SERVICE_OFFSET + 8)
#define WFS_SRVE_CEU_DEVICEPOSITION     (CEU_SERVICE_OFFSET + 9)
#define WFS_SRVE_CEU_POWER_SAVE_CHANGE  (CEU_SERVICE_OFFSET + 10)
#define WFS_USRE_CEU_TONERTHRESHOLD     (CEU_SERVICE_OFFSET + 11)

/* values of WFSCEUSTATUS.fwDevice */

#define WFS_CEU_DEVONLINE                WFS_STAT_DEVONLINE
#define WFS_CEU_DEVOFFLINE              WFS_STAT_DEVOFFLINE
#define WFS_CEU_DEVPOWEROFF             WFS_STAT_DEVPOWEROFF

```

CWA 16926-14:2015 (E)

```
#define WFS_CEU_DEVNODEDEVICE WFS_STAT_DEVNODEDEVICE
#define WFS_CEU_DEVHWERROR WFS_STAT_DEVHWERROR
#define WFS_CEU_DEVUSERERROR WFS_STAT_DEVUSERERROR
#define WFS_CEU_DEVBUSY WFS_STAT_DEVBUSY
#define WFS_CEU_DEVFRAUDATTEMPT WFS_STAT_DEVFRAUDATTEMPT
#define WFS_CEU_DEVPOTENTIALFRAUD WFS_STAT_DEVPOTENTIALFRAUD

/* values of WFSCEUSTATUS.fwMedia */

#define WFS_CEU_MEDIAPRESENT (1)
#define WFS_CEU_MEDIANOTPRESENT (2)
#define WFS_CEU_MEDIAJAMMED (3)
#define WFS_CEU_MEDIANOTSUPP (4)
#define WFS_CEU_MEDIAUNKNOWN (5)
#define WFS_CEU_MEDIAENTERING (6)
#define WFS_CEU_MEDIATOPPER (7)
#define WFS_CEU_MEDIAINHOPPER (8)
#define WFS_CEU_MEDIAOUTHOPPER (9)
#define WFS_CEU_MEDIAMSRE (10)
#define WFS_CEU_MEDIARETAINED (11)
#define WFS_CEU_MEDIAREMOVED (12)

/* values of WFSCEUSTATUS.fwRetainBin */

#define WFS_CEU_RETAINBINOK (1)
#define WFS_CEU_RETAINBINFULL (2)
#define WFS_CEU_RETAINBINHIGH (3)
#define WFS_CEU_RETAINBINNOTSUPP (4)

/* values of WFSCEUSTATUS.fwOutputBin */

#define WFS_CEU_OUTPUTBINOK (1)
#define WFS_CEU_OUTPUTBINFULL (2)
#define WFS_CEU_OUTPUTBINHIGH (3)
#define WFS_CEU_OUTPUTNOTSUPP (4)

/* values of WFSCEUSTATUS.fwInputBin */

#define WFS_CEU_INPUTBINOK (1)
#define WFS_CEU_INPUTBINEMPTY (2)
#define WFS_CEU_INPUTBINLOW (3)
#define WFS_CEU_INPUTNOTSUPP (4)

/* values of WFSCEUSTATUS.wDevicePosition
   WFSCEUDEVICEPOSITION.wPosition */

#define WFS_CEU_DEVICEINPOSITION (0)
#define WFS_CEU_DEVICENOTINPOSITION (1)
#define WFS_CEU_DEVICEPOSUNKNOWN (2)
#define WFS_CEU_DEVICEPOSNOTSUPP (3)

/* values of WFSCEUSTATUS.wToner */

#define WFS_CEU_TONERFULL (1)
#define WFS_CEU_TONERLOW (2)
#define WFS_CEU_TONEROUT (3)
#define WFS_CEU_TONERNOTSUPP (4)
#define WFS_CEU_TONERUNKNOWN (5)

/* values of WFSCEUCAPS.fwCharSupport,
   WFSCEUFORM.fwCharSupport */

#define WFS_CEU_ASCII (0x0001)
#define WFS_CEU_UNICODE (0x0002)

/* values of WFSCEUCAPS.fwType */

#define WFS_CEU_EMOSS (0x0001)
#define WFS_CEU_PRINT (0x0002)
```

```

/* values of WFSCEUFRMMEDIA.wBase */

#define WFS_CEU_INCH (1)
#define WFS_CEU_MM (2)
#define WFS_CEU_ROWCOLUMN (3)

/* values of WFSCEUFRMMEDIA.fwMediaType */

#define WFS_CEU_MEDIAECARD (1)
#define WFS_CEU_MEDIAPCARD (2)

/* values of WFSCEUFRMFIELD.fwType */

#define WFS_CEU_FIELDTEXT (1)
#define WFS_CEU_FIELDOCR (2)

/* values of WFSCEUFRMFIELD.fwClass */

#define WFS_CEU_CLASSSTATIC (1)
#define WFS_CEU_CLASSOPTIONAL (2)
#define WFS_CEU_CLASSREQUIRED (3)

/* values WFSCEUFIELDFAIL.wFailure */

#define WFS_CEU_FIELDREQUIRED (1)
#define WFS_CEU_FIELDSTATICOVWR (2)
#define WFS_CEU_FIELDOVERFLOW (3)
#define WFS_CEU_FIELDNOTFOUND (4)
#define WFS_CEU_FIELDNOTREAD (5)
#define WFS_CEU_FIELDNOTWRITE (6)
#define WFS_CEU_FIELDHWERROR (7)
#define WFS_CEU_FIELDTYPENOTSUPPORTED (8)
#define WFS_CEU_CHARSETFORM (9)

/* values of WFSCEUEMBOSSCARD.fwChipProtocols
   WFSCEUEMBOSSCARDEX.fwChipProtocols */

#define WFS_CEU_NOTSUPP (0x0000)
#define WFS_CEU_CHIPT0 (0x0001)
#define WFS_CEU_CHIPT1 (0x0002)
#define WFS_CEU_CHIP_PROTOCOL_NOT_REQUIRED (0x0004)

/* values of WFSCEUSUPPLYREPLEN.fwSupplyReplen */

#define WFS_CEU_REPLEN_TONER (0x0001)
#define WFS_CEU_REPLEN_INPUTBIN (0x0002)

/* WFS_EXEE_CEU_EMOSS_FAILURE Flags */

#define WFS_CEU_STEPPER_ERROR (1)
#define WFS_CEU_TOPPER_FOIL_BREAK (2)
#define WFS_CEU_CARD_FEED_ERROR (3)
#define WFS_CEU_MAGNETIC_STRIPE_ERROR (4)
#define WFS_CEU_RETAIN_BIN_FULL (5)
#define WFS_CEU_OUTPUT_BIN_FULL (6)
#define WFS_CEU_COVER_OPEN (7)
#define WFS_CEU_TOPPER_JAM (8)
#define WFS_CEU_STACKER_ERROR (9)
#define WFS_CEU_SYSTEM_ERROR (10)
#define WFS_CEU_OCR_ERROR (11)
#define WFS_CEU_EMOSS_LIMITS_EXCEEDED (12)
#define WFS_CEU_COMMUNICATIONS_FAILURE (13)
#define WFS_CEU_DATA_FORMAT_ERROR (14)
#define WFS_CEU_BUFFER_OVERRUN (15)
#define WFS_CEU_PRE_ENCODE_READ_ERROR (16)
#define WFS_CEU_PRE_ENCODE_DATA_MATCH_ERROR (17)
#define WFS_CEU_INPUT_BIN_EMPTY (18)
#define WFS_CEU_DEVICE_BUSY (19)
#define WFS_CEU_TONER_OUT_ERROR (20)
#define WFS_CEU_MEDIA_JAM (21)

```

```

/* values of lpwCeuMediacontrol parameter of WFS_CMD_CEU_RESET command */

#define WFS_CEU_CTRLTOINPUTBIN (1)
#define WFS_CEU_CTRLTOOUTPUTBIN (2)
#define WFS_CEU_CTRLTORETAINBIN (3)

/* WOSA/XFS CEU Errors */

#define WFS_ERR_CEU_FORMNOTFOUND (- (CEU_SERVICE_OFFSET + 1))
#define WFS_ERR_CEU_FORMINVALID (- (CEU_SERVICE_OFFSET + 2))
#define WFS_ERR_CEU_MEDIANOTFOUND (- (CEU_SERVICE_OFFSET + 3))
#define WFS_ERR_CEU_MEDIAINVALID (- (CEU_SERVICE_OFFSET + 4))
#define WFS_ERR_CEU_NOMEDIA (- (CEU_SERVICE_OFFSET + 5))
#define WFS_ERR_CEU_MEDIAOVERFLOW (- (CEU_SERVICE_OFFSET + 6))
#define WFS_ERR_CEU_IDC_FORMNOTFOUND (- (CEU_SERVICE_OFFSET + 7))
#define WFS_ERR_CEU_IDC_FORMINVALID (- (CEU_SERVICE_OFFSET + 8))
#define WFS_ERR_CEU_INVALIDDATA (- (CEU_SERVICE_OFFSET + 9))
#define WFS_ERR_CEU_PROTOCOLNOTSUPP (- (CEU_SERVICE_OFFSET + 10))
#define WFS_ERR_CEU_ATRNOTOBTAINED (- (CEU_SERVICE_OFFSET + 11))
#define WFS_ERR_CEU_FIELDSPECFAILURE (- (CEU_SERVICE_OFFSET + 12))
#define WFS_ERR_CEU_FIELDERROR (- (CEU_SERVICE_OFFSET + 13))
#define WFS_ERR_CEU_EMBOSFAILURE (- (CEU_SERVICE_OFFSET + 14))
#define WFS_ERR_CEU_FIELDNOTFOUND (- (CEU_SERVICE_OFFSET + 15))
#define WFS_ERR_CEU_POWERSAVETOOSHORT (- (CEU_SERVICE_OFFSET + 16))
#define WFS_ERR_CEU_POWERSAVEMEDIAPRESENT (- (CEU_SERVICE_OFFSET + 17))
#define WFS_ERR_CEU_CHARSETDATA (- (CEU_SERVICE_OFFSET + 18))
#define WFS_ERR_CEU_COMMANDUNSUPP (- (CEU_SERVICE_OFFSET + 19))
#define WFS_ERR_CEU_SYNCHRONIZEUNSUPP (- (CEU_SERVICE_OFFSET + 20))

/* values of WFSCEUSTATUS.wAntiFraudModule */

#define WFS_CEU_AFMNOTSUPP (0)
#define WFS_CEU_AFMOK (1)
#define WFS_CEU_AFMINOP (2)
#define WFS_CEU_AFMDEVICEDETECTED (3)
#define WFS_CEU_AFMUNKNOWN (4)

/*=====*/
/* CEU Info Command Structures and variables */
/*=====*/

typedef struct _wfs_ceu_status
{
    WORD fwDevice;
    WORD fwMedia;
    WORD fwRetainBin;
    WORD fwOutputBin;
    WORD fwInputBin;
    USHORT usTotalCards;
    USHORT usOutputCards;
    USHORT usRetainCards;
    LPSTR lpszExtra;
    WORD wDevicePosition;
    USHORT usPowerSaveRecoveryTime;
    WORD wToner;
    WORD wAntiFraudModule;
} WFSCEUSTATUS, *LPWFSCEUSTATUS;

typedef struct _wfs_ceu_caps
{
    WORD wClass;
    BOOL bCompound;
    BOOL bCompareMagneticStripe;
    BOOL bMagneticStripeRead;
    BOOL bMagneticStripeWrite;
    BOOL bChipIO;
    WORD wChipProtocol;
}

```

```

    LPSTR          lpszExtra;
    BOOL           bPowerSaveControl;
    WORD           fwCharSupport;
    WORD           fwType;
    BOOL           bAntiFraudModule;
    LPDWORD        lpdwSynchronizableCommands;
} WFSCEUCAPS, *LPWFSCEUCAPS;

typedef struct _wfs_ceu_form
{
    LPSTR          lpszFormName;
    LPSTR          lpszFields;
    WORD           fwCharSupport;
    WORD           wLanguageID;
} WFSCEUFORM, *LPWFSCEUFORM;

typedef struct _wfs_ceu_frm_media
{
    WORD           fwMediaType;
    WORD           wBase;
    WORD           wUnitX;
    WORD           wUnitY;
    WORD           wSizeWidth;
    WORD           wSizeHeight;
    WORD           wEmbossAreaX;
    WORD           wEmbossAreaY;
    WORD           wEmbossAreaWidth;
    WORD           wEmbossAreaHeight;
    WORD           wRestrictedAreaX;
    WORD           wRestrictedAreaY;
    WORD           wRestrictedAreaWidth;
    WORD           wRestrictedAreaHeight;
} WFSCEUFRMMEDIA, *LPWFSCEUFRMMEDIA;

typedef struct _wfs_ceu_query_field
{
    LPSTR          lpszFormName;
    LPSTR          lpszFieldName;
} WFSCEUQUERYFIELD, *LPWFSCEUQUERYFIELD;

typedef struct _wfs_ceu_frm_field
{
    LPSTR          lpszFieldName;
    WORD           fwType;
    WORD           fwClass;
    LPSTR          lpszInitialValue;
    LPSTR          lpszFormat;
    LPWSTR         lpszUNICODEInitialValue;
    LPWSTR         lpszUNICODEFormat;
    WORD           wLanguageID;
} WFSCEUFRMFIELD, *LPWFSCEUFRMFIELD;

/*=====*/
/* CEU Execute Command Structures */
/*=====*/

typedef struct _wfs_ceu_emboss_card
{
    LPSTR          lpszFormName;
    LPSTR          lpszMediaName;
    LPSTR          lpszFields;
    LPSTR          lpszCompareFormIOFormName;
    LPSTR          lpszCompareFormIOTrackData;
    LPSTR          lpszFormIOFormName;
    LPSTR          lpszFormIOTrackData;
    WORD           wChipProtocol;
    ULONG          ulChipDataLength;
    LPBYTE         lpbChipData;
} WFSCEUEMBOSSCARD, *LPWFSCEUEMBOSSCARD;

```

```

typedef struct _wfs_ceu_power_save_control
{
    USHORT                usMaxPowerSaveRecoveryTime;
} WFSCEUPOWERSAVECONTROL, *LPWFSCEUPOWERSAVECONTROL;

typedef struct _wfs_ceu_emboss_card_ex
{
    LPSTR                lpszFormName;
    LPSTR                lpszMediaName;
    LPSTR                lpszFields;
    LPSTR                lpszCompareFormIOFormName;
    LPSTR                lpszCompareFormIOTrackData;
    LPSTR                lpszFormIOFormName;
    LPSTR                lpszFormIOTrackData;
    WORD                wChipProtocol;
    ULONG                ulChipDataLength;
    LPBYTE               lpbChipData;
    LPWSTR               lpszUNICODEFields;
} WFSCEUEMBOSSCARDEX, *LPWFSCEUEMBOSSCARDEX;

typedef struct _wfs_ceu_supply_replen
{
    WORD                fwSupplyReplen;
} WFSCEUSUPPLYREPLEN, *LPWFSCEUSUPPLYREPLEN;

typedef struct _wfs_ceu_synchronize_command
{
    DWORD                dwCommand;
    LPVOID               lpCmdData;
} WFSCEUSYNCHRONIZECOMMAND, *LPWFSCEUSYNCHRONIZECOMMAND;

/*=====*/
/* CEU Message Structures */
/*=====*/

typedef struct _wfs_ceu_field_failure
{
    LPSTR                lpszFormName;
    LPSTR                lpszFieldName;
    WORD                wFailure;
} WFSCEUFIELDFAIL, *LPWFSCEUFIELDFAIL;

typedef struct _wfs_ceu_device_position
{
    WORD                wPosition;
} WFSCEUDEVICEPOSITION, *LPWFSCEUDEVICEPOSITION;

typedef struct _wfs_ceu_power_save_change
{
    USHORT                usPowerSaveRecoveryTime;
} WFSCEUPOWERSAVECHANGE, *LPWFSCEUPOWERSAVECHANGE;

typedef struct _wfs_ceu_toner_status
{
    LPWORD               lpwTonerThreshold;
} WFSCEUTONERSTATUS, *LPWFSCEUTONERSTATUS;

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

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

#endif /* __INC_XFSCEU__H */

```