 Extensions for Financial Services (XFS) interface specification
Release 3.20 - 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, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and United Kingdom.
# Table of Contents

Foreword ....................................................................................................................... 4  
1. Introduction .......................................................................................................... 7  
   1.1 Background to Release 3.20 ........................................................................ 7  
   1.2 XFS Service-Specific Programming .............................................................. 7  
2. Card Embossing Units ........................................................................................ 8  
3. References ........................................................................................................... 9  
4. Info Commands ................................................................................................. 10  
   4.1 WFS-INF-CEU-STATUS ............................................................................... 10  
   4.2 WFS-INF-CEU-CAPABILITIES ................................................................. 14  
   4.3 WFS-INF-CEU-FORM_LIST ..................................................................... 16  
   4.4 WFS-INF-CEU-MEDIA_LIST ................................................................. 17  
   4.5 WFS-INF-CEU-QUERY_FORM ............................................................... 18  
   4.6 WFS-INF-CEU-QUERY_MEDIA ............................................................... 19  
   4.7 WFS-INF-CEU-QUERY_FIELD ............................................................... 21  
5. Execute Commands ............................................................................................ 23  
   5.1 WFS-CMD-CEU-EMBOSS_CARD ......................................................... 23  
   5.2 WFS-CMD-CEU-RESET ........................................................................... 25  
   5.3 WFS-CMD-CEU-POWER_SAVE_CONTROL .......................................... 26  
   5.4 WFS-CMD-CEU-EMBOSS_CARD_EX .................................................... 27  
   5.5 WFS-CMD-CEU-SUPPLY_REPLENISH ................................................. 30  
6. Events .................................................................................................................. 31  
   6.1 WFS-SRVE-CEU-INPUTBINTHRESHOLD ............................................. 31  
   6.2 WFS-SRVE-CEU-OUTPUTBINTHRESHOLD .......................................... 32  
   6.3 WFS-SRVE-CEU-RETAINBINTHRESHOLD ........................................... 33  
   6.4 WFS-EXEE-CEU-FIELDERROR ............................................................ 34  
   6.5 WFS-EXEE-CEU-FIELDWARNING ....................................................... 35  
   6.6 WFS-SRVE-CEU-MEDIAREMOVED .................................................... 36  
   6.7 WFS-SRVE-CEU-MEDIADETECTED .................................................... 37  
   6.8 WFS-EXEE-CEU-EMBOSS_FAILURE .................................................... 38  
   6.9 WFS-SRVE-CEU-DEVICEPOSITION ................................................... 39  
   6.10 WFS-SRVE-CEU-POWER_SAVE_CHANGE ....................................... 40  
   6.11 WFS-USRE-CEU-TONERTHRESHOLD .................................................. 41  
7. Embossing Form, Field and Media Definitions ............................................. 42  
   7.1 Definition Syntax ...................................................................................... 42  
   7.2 Embossing Form and Media Measurements ........................................... 43  
   7.3 Embossing Form Definition ....................................................................... 44
7.4 Embossing Field Definition

7.5 Media Definition

8. C-Header file
Foreword

This CWA is revision 3.20 of the XFS interface specification.

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties on 2011-06-29, 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 3.20.

A list of the individuals and organizations which supported the technical consensus represented by the CEN Workshop Agreement is available to purchasers from the CEN-CENELEC Management Centre. These organizations were drawn from the banking sector. The CEN/ISSS 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 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
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 Device 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 Class
Part 44: XFS MIB Application Management
Part 45: XFS MIB Device Specific Definitions - Card Dispenser Device Class
Part 46: XFS MIB Device Specific Definitions - Barcode Reader Device Class
Part 47: XFS MIB Device Specific Definitions - Item Processing Module Device Class
Parts 48 - 60 are reserved for future use.
Part 61: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 62: Printer and Scanning Device Class Interface - Migration from Version 3.10 (CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 63: Identification Card Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 65: PIN Keypad Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 67: Depository Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 71: Camera Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 72: Alarm Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 74: Cash-In Module Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 75: Card Dispenser Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 76: Barcode Reader Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 77: Item Processing Module Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (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/cen/pages/default.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/ISSS 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 2011-06-23 and was successfully closed on 2011-07-23. The final text of this CWA was submitted to CEN for publication on 2011-08-26.

This CEN Workshop Agreement is publicly available as a reference document from the National Members of CEN: 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, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland 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:

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.00</td>
<td>October 18, 2000</td>
<td>Initial release.</td>
</tr>
<tr>
<td>3.10</td>
<td>November 29, 2007</td>
<td>For a description of changes from version 3.00 to version 3.10 see the CEU 3.10 Migration document.</td>
</tr>
<tr>
<td>3.20</td>
<td>March 2nd, 2011</td>
<td>For a description of changes from version 3.10 to version 3.20 see the CEU 3.20 Migration document.</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Background to Release 3.20

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

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

Release 3.20 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 does not include any new device classes. Notable major enhancements include Mixed Media processing to allow mixed cash and check accepting, as well as the addition of new commands to the CIM, PTR and IDC to allow better support of the Japanese marketplace.

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 WFSEexecute 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, WFSEexecute, 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

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer’s Reference Revision 3.20
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_DEVONLINE</td>
<td>The device is present, powered on and online (i.e. operational, not busy processing a request and not in an error state).</td>
</tr>
<tr>
<td>WFS_CEU_DEVOFFLINE</td>
<td>The device is offline (e.g. the operator has taken the device offline by turning a switch or pulling out the device).</td>
</tr>
<tr>
<td>WFS_CEU_DEVPOWEROFF</td>
<td>The device is powered off or physically not connected.</td>
</tr>
<tr>
<td>WFS_CEU_DEVNODEVICE</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_CEU_DEVHWERROR</td>
<td>The device is present but inoperable due to a hardware fault that prevents it from being used.</td>
</tr>
<tr>
<td>WFS_CEU_DEVUSERERROR</td>
<td>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).</td>
</tr>
<tr>
<td>WFS_CEU_DEVBUSY</td>
<td>The device is busy and unable to process an execute command at this time.</td>
</tr>
<tr>
<td>WFS_CEU_DEVFRAUDATTEMPT</td>
<td>The device is present but is inoperable because it has detected a fraud attempt.</td>
</tr>
</tbody>
</table>
WFS_CEU_DEV_POTENTIAL_FRAUD

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_MEDIAPRESENT</td>
<td>Media is present in the device, not in the entering position and not jammed.</td>
</tr>
<tr>
<td>WFS_CEU_MEDIANOTPRESENT</td>
<td>Media is not present in the device and not at the entering position.</td>
</tr>
<tr>
<td>WFS_CEU_MEDIAJAMMED</td>
<td>Media is jammed in the device; operator intervention is required.</td>
</tr>
<tr>
<td>WFS_CEU_MEDIANOTSUPP</td>
<td>Capability to report media position is not supported by the device.</td>
</tr>
<tr>
<td>WFS_CEU_MEDIAUNKNOWN</td>
<td>The media state cannot be determined with the device in its current state (e.g. the value of fwDevice is WFS_CEU_DEV_NODEVICE, WFS_CEU_DEV_POWEROFF, WFS_CEU_DEV_OFFLINE, or WFS_CEU_DEV_HWERROR).</td>
</tr>
<tr>
<td>WFS_CEU_MEDIAENTERING</td>
<td>Media is at the entry/exit slot.</td>
</tr>
<tr>
<td>WFS_CEU_MEDIATOPPER</td>
<td>Topper failure.</td>
</tr>
<tr>
<td>WFS_CEU.MEDIAINHOPPER</td>
<td>Card is positioned in input bin.</td>
</tr>
<tr>
<td>WFS_CEU.MEDIAOUTHOPPER</td>
<td>Card is positioned in output bin.</td>
</tr>
<tr>
<td>WFS_CEU.MEDIA_MSRE</td>
<td>Encoding failure.</td>
</tr>
<tr>
<td>WFS_CEU.MEDIA_RETAI NED</td>
<td>Card is positioned in retain bin.</td>
</tr>
</tbody>
</table>

fwRetainBin

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_RETAINBINOK</td>
<td>The retain bin is in a good state.</td>
</tr>
<tr>
<td>WFS_CEU_RETAINBINFULL</td>
<td>The retain bin is full.</td>
</tr>
<tr>
<td>WFS_CEU_RETAINBINHIGH</td>
<td>The retain bin is nearly full.</td>
</tr>
<tr>
<td>WFS_CEU_RETAINBINNOTSUPP</td>
<td>The retain bin state can not be reported.</td>
</tr>
</tbody>
</table>

fwOutputBin

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_OUTPUTBINOK</td>
<td>The output bin is in a good state.</td>
</tr>
<tr>
<td>WFS_CEU_OUTPUTBINFULL</td>
<td>The output bin is full.</td>
</tr>
<tr>
<td>WFS_CEU_OUTPUTBINHIGH</td>
<td>The output bin is nearly full.</td>
</tr>
<tr>
<td>WFS_CEU_OUTPUTNOTSUPP</td>
<td>The output bin state can not be reported.</td>
</tr>
</tbody>
</table>

fwInputBin

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_INPUTBINOK</td>
<td>The input bin is in a good state.</td>
</tr>
<tr>
<td>WFS_CEU_INPUTBINEMPTY</td>
<td>The input bin is empty.</td>
</tr>
<tr>
<td>WFS_CEU_INPUTBINLOW</td>
<td>The input bin is nearly empty.</td>
</tr>
<tr>
<td>WFS_CEU_INPUTNOTSUPP</td>
<td>The input bin state can not be reported.</td>
</tr>
</tbody>
</table>

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_DEVICESMNOTINPOSITION, 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 (e.g. WFS_CEU_DEVICEINPOSITION) then media may not be presented through the normal customer interface. This value is one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_DEVICEINPOSITION</td>
<td>The device is in its normal operating position, or is fixed in place and cannot be moved.</td>
</tr>
<tr>
<td>WFS_CEU_DEVICESMNOTINPOSITION</td>
<td>The device has been removed from its normal operating position.</td>
</tr>
<tr>
<td>WFS_CEU_DEVICEPOSUNKNOWN</td>
<td>Due to a hardware error or other condition, the position of the device cannot be determined.</td>
</tr>
<tr>
<td>WFS_CEU_DEVICEPOSNOTSUPP</td>
<td>The physical device does not have the capability of detecting the position.</td>
</tr>
</tbody>
</table>

**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:

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

**wAntiFraudModule**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_AFMNOTSUPP</td>
<td>No anti-fraud module is available.</td>
</tr>
<tr>
<td>WFS_CEU_AFMOK</td>
<td>Anti-fraud module is in a good state and no foreign device is detected.</td>
</tr>
<tr>
<td>WFS_CEU_AFMINOP</td>
<td>Anti-fraud module is inoperable.</td>
</tr>
<tr>
<td>WFS_CEU_AFMDVICEDETECTED</td>
<td>Anti-fraud module detected the presence of a foreign device.</td>
</tr>
<tr>
<td>WFS_CEU_AFMUNKNOWN</td>
<td>The state of the anti-fraud module cannot be determined.</td>
</tr>
</tbody>
</table>

**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 \textit{fwDevice} field will report 
\texttt{WFS\_CEU\_DEVPOWEROFF} when the device has been removed or 
\texttt{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;
} 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_NOTSUPP</td>
<td>The CEU card unit can not handle chip cards.</td>
</tr>
<tr>
<td>WFS_CEU_CHIPT0</td>
<td>The CEU card unit can handle the T=0 protocol.</td>
</tr>
<tr>
<td>WFS_CEU_CHIPT1</td>
<td>The CEU card unit can handle the T=1 protocol.</td>
</tr>
<tr>
<td>WFS_CEU_CHIP_PROTOCOL_NOT_REQUIRED</td>
<td>The CEU card unit is capable of communicating with a chip card without requiring the application to specify any protocol.</td>
</tr>
</tbody>
</table>

- **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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_ASCII</td>
<td>ASCII is supported for XFS forms.</td>
</tr>
<tr>
<td>WFS_CEU_UNICODE</td>
<td>UNICODE is supported for XFS forms.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_EMBOSS</td>
<td>The CEU card unit supports embossing data on cards.</td>
</tr>
<tr>
<td>WFS_CEU_PRINT</td>
<td>The CEU card unit supports printing data on cards.</td>
</tr>
</tbody>
</table>

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

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.
4.3 **WFS-INF_CEU_FORM_LIST**

<table>
<thead>
<tr>
<th>Description</th>
<th>This command is used to retrieve the list of forms available on the device.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input Param</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Output Param</strong></td>
<td>LPSTR lpszFormList;</td>
</tr>
</tbody>
</table>

- *lpszFormList*
  - 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

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

lpszFormName
Points to the null-terminated form name on which to retrieve details.

Output Param
LPWFSCEUFORM lpForm;

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

lpszFormName
Specifies the null-terminated name of the form.

lpszFields
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_ASCII</td>
<td>ASCII is supported for XFS forms initial data values and FORMAT strings.</td>
</tr>
<tr>
<td>WFS_CEU_UNICODE</td>
<td>UNICODE is supported for XFS forms initial data values and FORMAT strings.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CEU_FORMNOTFOUND</td>
<td>The specified form cannot be found.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_FORMINVALID</td>
<td>The specified form is invalid.</td>
</tr>
</tbody>
</table>

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;

```c
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_MEDIAECARD</td>
<td>Embossable card media.</td>
</tr>
<tr>
<td>WFS_CEU_MEDIAPCARD</td>
<td>Printable card media.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_INCH</td>
<td>The base unit is inches.</td>
</tr>
<tr>
<td>WFS_CEU_MM</td>
<td>The base unit is millimeters.</td>
</tr>
<tr>
<td>WFS_CEU_ROWCOLUMN</td>
<td>The base unit is rows and columns.</td>
</tr>
</tbody>
</table>

- **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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CEU_MEDIANOTFOUND</td>
<td>The specified media definition cannot be found.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_MEDIAINVALID</td>
<td>The specified media definition is invalid.</td>
</tr>
</tbody>
</table>

**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    lpszFormName;
    LPSTR    lpszFieldName;
} WFSCEUQUERYFIELD, *LPWFSCEUQUERYFIELD;

lpszFormName
Points to the null-terminated form name.

lpszFieldName
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 lpszInitialValue or lpszUNICODEInitialValue 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    lpszFieldName;
    WORD    fwType;
    WORD    fwClass;
    LPSTR    lpszInitialValue;
    LPSTR    lpszFormat;
    LPWSTR   lpszUNICODEInitialValue;
    LPWSTR   lpszUNICODEFormat;
    WORD    wLanguageID;
} WFSCEUFRMFIELD, *LPWFSCEUFRMFIELD;

lpszFieldName
Pointer to the null-terminated field name.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_FIELDTEXT</td>
<td>A text field.</td>
</tr>
<tr>
<td>WFS_CEU_FIELDOCR</td>
<td>An Optical Character Recognition (OCR) field.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_CLASSSTATIC</td>
<td>The field data cannot be set by the application.</td>
</tr>
<tr>
<td>WFS_CEU_CLASSOPTIONAL</td>
<td>The field data can be set by the application.</td>
</tr>
<tr>
<td>WFS_CEU_CLASSREQUIRED</td>
<td>The field data must be set by the application.</td>
</tr>
</tbody>
</table>

lpszInitialValue
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.

lpszFormat
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.

lpszUNICODEInitialValue
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CEU_FORMNOTFOUND</td>
<td>The specified form cannot be found.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_FIELDNOTFOUND</td>
<td>The specified field cannot be found.</td>
</tr>
</tbody>
</table>

**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    lpszMediaName;
    LPSTR    lpszFields;
    LPSTR    lpszCompareFormIOFormName;
    LPSTR    lpszCompareFormIOTrackData;
    LPSTR    lpszFormIOFormName;
    LPSTR    lpszFormIOTrackData;
    WORD    wChipProtocol;
    ULONG    ulChipDataLength;
    LPBYTE   lpbChipData;
} WFSCEUEMBOSSCARD, *LPWFSCEUEMBOSSCARD;

lpszFormName
Pointer to the null-terminated form name.

lpszMediaName
Pointer to the null-terminated media name.

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

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_CHIPT0</td>
<td>Use the T=0 protocol to communicate with the chip.</td>
</tr>
<tr>
<td>WFS_CEU_CHIPT1</td>
<td>Use the T=1 protocol to communicate with the chip.</td>
</tr>
</tbody>
</table>
WFS_CEU_CHIP_PROTOCOL_NOT_REQUIRED

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

\textit{ulChipDataLength}

Specifies the length of the following field \textit{lpbChipData}.

\textit{lpbChipData}

Points to the data sent to the chip.

\textbf{Output Param}

None.

\textbf{Error Codes}

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CEU_FORMNOTFOUND</td>
<td>The specified form definition cannot be found.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_FORMINVALID</td>
<td>The specified form definition is invalid.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_MEDIANOTFOUND</td>
<td>The specified media definition cannot be found.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_MEDIAINVALID</td>
<td>The specified media definition is invalid.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_MEDIANOVERFLOW</td>
<td>The form overflowed the media.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_IDC_FORMNOTFOUND</td>
<td>The specified IDC form definition cannot be found.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_IDC_FORMINVALID</td>
<td>The specified IDC form definition is invalid.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_INVALIDDATA</td>
<td>An error occurred while communicating with the chip.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_PROTOCOLNOTSUPP</td>
<td>The protocol used was not supported by the Service Provider.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_ATRNOTOBTAINED</td>
<td>The ATR was not obtained by issuing the IDC class \textit{WFS_CMD_CEU_READ_RAW_DATA} command.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_FIELDSPECFAILURE</td>
<td>The syntax of the \textit{lpszFields} member is invalid.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_FIELDERROR</td>
<td>An error occurred while processing a field, causing termination of the emboss request. An execute event \textit{WFS_EXEE_CEU_FIELDERROR} is posted with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_EMBOSSFAILURE</td>
<td>A card embossing failure has occurred.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_EMBOSS_FAILURE</td>
<td>A failure has occurred during Emboss processing. A service event \textit{WFS_EXEE_CEU_EMBOS FAILURE} is posted with details.</td>
</tr>
</tbody>
</table>

\textbf{Events}

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

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

\textbf{Comments}

This command is only supported for backwards compatibility; the
\textit{WFS_CMD_CEU_EMBOS 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 lpwCeuParam;

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_CTRLTOINPUTBIN</td>
<td>Any media detected should be moved to the input bin.</td>
</tr>
<tr>
<td>WFS_CEU_CTRLTOOUTPUTBIN</td>
<td>Any media detected should be moved to the output bin.</td>
</tr>
<tr>
<td>WFS_CEU_CTRLTORETAUBIN</td>
<td>Any media detected should be moved to the retain bin.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_SRVE_CEU_OUTPUTBINTHRESHOLD</td>
<td>Output bin is nearly full.</td>
</tr>
<tr>
<td>WFS_SRVE_CEU_RETAUBINTHRESHOLD</td>
<td>Retain bin is nearly full.</td>
</tr>
<tr>
<td>WFS_SRVE_CEU.MEDIADETECTED</td>
<td>Media was detected in the device during a reset.</td>
</tr>
</tbody>
</table>

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

If *lpwCeuParam* 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CEU_POWERSAVETOOSHORT</td>
<td>The power saving mode has not been activated because the device is not able to resume from the power saving mode within the specified usMaxPowerSaveRecoveryTime value.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_POWERSAVEMEDIAPRESENT</td>
<td>The power saving mode has not been activated because media is present inside the device.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_SRVE_CEU_POWER_SAVE_CHANGE</td>
<td>The power save recovery time has changed.</td>
</tr>
</tbody>
</table>

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>=<FieldValue>" 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>]=<FieldValue>", 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_CHIPT0</td>
<td>Use the T=0 protocol to communicate with the chip.</td>
</tr>
<tr>
<td>WFS_CEU_CHIPT1</td>
<td>Use the T=1 protocol to communicate with the chip.</td>
</tr>
</tbody>
</table>
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>=<FieldValue>" 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>]=<FieldValue>", 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CEU_FORMNOTFOUND</td>
<td>The specified form definition cannot be found.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_FORMINVALID</td>
<td>The specified form definition is invalid.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_MEDIANOTFOUND</td>
<td>The specified media definition cannot be found.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_MEDIAINVALID</td>
<td>The specified media definition is invalid.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_NOMEDIA</td>
<td>There is no card inside the device.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_MEDIAOVERFLOW</td>
<td>The form overflowed the media.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_IDC_FORMNOTFOUND</td>
<td>The specified IDC form definition cannot be found.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_IDC_FORMINVALID</td>
<td>The specified IDC form definition is invalid.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_INVALIDDATA</td>
<td>An error occurred while communicating with the chip.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_PROTOCOLNOTSUPP</td>
<td>The protocol used was not supported by the service provider.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_ATRNOTOBTAINED</td>
<td>The ATR was not obtained by issuing the IDC class</td>
</tr>
<tr>
<td></td>
<td>WFS_CMD_IDC_READ_RAW_DATA command.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_FIELDSPECFAILURE</td>
<td>The syntax of the lpszFields member is invalid.</td>
</tr>
<tr>
<td>WFS_ERR_CEUFIELDERROR</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_EMMROSSFAILURE</td>
<td>A failure has occurred during Emboss or Print processing. A service event WFS_EXEE_CEU_EMMROSS_FAILURE is posted with details.</td>
</tr>
<tr>
<td>WFS_ERR_CEU_CHARSETDATA</td>
<td>The character set(s) supported by the Service Provider is inconsistent with the use of the lpszFields or lpszUNICODEFields fields.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_SRVE_CEU_INPUTBINTHRESHOLD</td>
<td>Input bin is nearly empty.</td>
</tr>
<tr>
<td>WFS_SRVE_CEU_OUTPUTBINTHRESHOLD</td>
<td>Output bin is nearly full.</td>
</tr>
</tbody>
</table>
WFS_SRVE_CEU_RETAINBINTHRESHOLD  Retain bin is nearly full.
WFS_EXEE_CEU_EMBOSS_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 lpszFields or lpszUNICODEFields as an input parameter, depending upon the Service Provider capabilities. Legacy (non-UNICODE aware) applications will only use the lpszFields field. UNICODE applications can use either the lpszFields or lpszUNICODEFields fields, provided the Service Provider is UNICODE compliant.
5.5 WFS_CMD_CEU_SUPPLY_REPLENISH

**Description**

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.

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.

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.

**Input Param**

LPWFSCEUSUPPLYREPLEN lpSupplyReplen;

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

fwSupplyReplen

Specifies the supply that was replenished as a combination of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<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>
</tbody>
</table>

**Comments**

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.
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_INPUTBINOK</td>
<td>The input bin is in a good state.</td>
</tr>
<tr>
<td>WFS_CEU_INPUTBINLOW</td>
<td>The input bin is nearly empty.</td>
</tr>
<tr>
<td>WFS_CEU_INPUTBINEMPTY</td>
<td>The input bin is empty.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_OUTPUTBINOK</td>
<td>The output bin is in a good state.</td>
</tr>
<tr>
<td>WFS_CEU_OUTPUTBINFULL</td>
<td>The output bin is full.</td>
</tr>
<tr>
<td>WFS_CEU_OUTPUTBINHIGH</td>
<td>The output bin is nearly full.</td>
</tr>
</tbody>
</table>

Comments
None.
**6.3 WFS_SRVE_CEU_RETAINBIN_THRESHOLD**

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_RETAINBINOK</td>
<td>The retain bin is in a good state.</td>
</tr>
<tr>
<td>WFS_CEU_RETAINBINFULL</td>
<td>The retain bin is full.</td>
</tr>
<tr>
<td>WFS_CEU_RETAINBINHIGH</td>
<td>The retain bin is nearly full.</td>
</tr>
</tbody>
</table>

**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    lpszFormName;
    LPSTR    lpszFieldName;
    WORD     wFailure;
} WFSCEUFIELDFAIL, *LPWFSCEUFIELDFAIL;

*lpszFormName*  
Points to the null-terminated form name.

*lpszFieldName*  
Points to the null-terminated field name.

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

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

**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

<table>
<thead>
<tr>
<th>Description</th>
<th>This event is generated when a card is removed before completion of a write operation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Param</td>
<td>None.</td>
</tr>
<tr>
<td>Comments</td>
<td>None.</td>
</tr>
</tbody>
</table>
## 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_MEDIARETAINED</td>
<td>The media was successfully retained during the reset operation.</td>
</tr>
<tr>
<td>WFS_CEU_MEDIAREMOVED</td>
<td>The media was removed during the reset operation.</td>
</tr>
<tr>
<td>WFS_CEU_MEDIAJAMMED</td>
<td>The media is jammed in the device.</td>
</tr>
<tr>
<td>WFS_CEU_MEDIAUNKNOWN</td>
<td>The media is in an unknown position.</td>
</tr>
</tbody>
</table>

**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:

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

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_DEVICEINPOSITION</td>
<td>The device is in its normal operating position.</td>
</tr>
<tr>
<td>WFS_CEU_DEVICENOTINPOSITION</td>
<td>The device has been removed from its normal operating position.</td>
</tr>
<tr>
<td>WFS_CEUDEVICEPOSUNKNOWN</td>
<td>The position of the device cannot be determined.</td>
</tr>
</tbody>
</table>

**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
LPWFSCEUPOWERSAVECHANGE lpPowerSaveChange;

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

usPowerSaveRecoveryTime
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.

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**

```c
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CEU_TONERFULL</td>
<td>The toner, ink or ribbon in the printer is in a good state.</td>
</tr>
<tr>
<td>WFS_CEU_TONERLOW</td>
<td>The toner or ink in the printer is low or the print contrast with a ribbon is weak.</td>
</tr>
<tr>
<td>WFS_CEU_TONEROUT</td>
<td>The toner or ink in the printer is out or the print contrast with a ribbon is not sufficient any more.</td>
</tr>
</tbody>
</table>

**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)
- EMBOSAREA (x, y, width and height values)
- RESTRICTED (x, y, width and height values)
## 7.3 Embossing Form Definition

<table>
<thead>
<tr>
<th>XFSFORM</th>
<th>formname</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN</td>
<td></td>
</tr>
<tr>
<td>(required) UNIT</td>
<td>base,</td>
</tr>
<tr>
<td></td>
<td>x,</td>
</tr>
<tr>
<td></td>
<td>y,</td>
</tr>
<tr>
<td>(required) SIZE</td>
<td>width,</td>
</tr>
<tr>
<td></td>
<td>height</td>
</tr>
<tr>
<td>ALIGNMENT</td>
<td>alignment,</td>
</tr>
<tr>
<td></td>
<td>xoffset,</td>
</tr>
<tr>
<td></td>
<td>yoffset</td>
</tr>
<tr>
<td>VERSION</td>
<td>major,</td>
</tr>
<tr>
<td></td>
<td>minor,</td>
</tr>
<tr>
<td></td>
<td>date*</td>
</tr>
<tr>
<td></td>
<td>author*</td>
</tr>
<tr>
<td>(required) LANGUAGE</td>
<td>languageID</td>
</tr>
<tr>
<td>COPYRIGHT</td>
<td>copyright*</td>
</tr>
<tr>
<td>TITLE</td>
<td>title*</td>
</tr>
<tr>
<td>COMMENT</td>
<td>comment*</td>
</tr>
<tr>
<td>USERPROMPT</td>
<td>prompt*</td>
</tr>
<tr>
<td>[ XFSFIELD</td>
<td>fieldName</td>
</tr>
<tr>
<td>BEGIN</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
</tr>
<tr>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>
### 7.4 Embossing Field Definition

<table>
<thead>
<tr>
<th>XFSFIELD</th>
<th>fieldname</th>
</tr>
</thead>
<tbody>
<tr>
<td>BEGIN</td>
<td></td>
</tr>
<tr>
<td>(required) POSITION x, y</td>
<td>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)</td>
</tr>
<tr>
<td>HPOSITION</td>
<td>Horizontal field positioning relative to: LEFT (default) RIGHT</td>
</tr>
<tr>
<td>VPOSITION</td>
<td>Vertical field positioning relative to: TOP BOTTOM (default)</td>
</tr>
<tr>
<td>SIDE</td>
<td>Side of card: FRONT (default) BACK</td>
</tr>
<tr>
<td>(required) SIZE width, height</td>
<td>Field width Field height</td>
</tr>
<tr>
<td>TYPE fieldtype</td>
<td>Type of field: TEXT (default) OCR</td>
</tr>
<tr>
<td>CLASS class</td>
<td>Field class OPTIONAL (default) STATIC REQUIRED</td>
</tr>
<tr>
<td>CASE case</td>
<td>Convert field contents to NOCHANGE (default) UPPER LOWER</td>
</tr>
<tr>
<td>HORIZONTAL justify</td>
<td>Horizontal alignment of field contents LEFT (default) RIGHT CENTER JUSTIFY</td>
</tr>
<tr>
<td>VERTICAL justify</td>
<td>Vertical alignment of field contents BOTTOM (default) CENTER TOP</td>
</tr>
<tr>
<td>FONT fontname*</td>
<td>Font name; in some cases this predefines the following parameters:</td>
</tr>
<tr>
<td>POINTSIZE pointsize</td>
<td>Point size</td>
</tr>
<tr>
<td>CPI cpi</td>
<td>Characters per inch</td>
</tr>
<tr>
<td>LPI lpi</td>
<td>Lines per inch</td>
</tr>
<tr>
<td>FORMAT formatstring*</td>
<td>This is an application defined input field describing how the application should format the data. This may be interpreted by the Service Provider.</td>
</tr>
<tr>
<td>INITIALVALUE value*</td>
<td>Initial value</td>
</tr>
<tr>
<td>LANGUAGE languageID</td>
<td>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.</td>
</tr>
<tr>
<td>END</td>
<td></td>
</tr>
</tbody>
</table>
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.

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

/******************************************************************************
*                                                                             *
stfsceu.h  XFS - Card Embossing Unit (CEU) definitions                      *
*                                                                             *
*             Version 3.20  (March 02 2011)                                   *
*                                                                             *
******************************************************************************/

#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       (0x1403) /* Version 3.20 */
#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_EMBOSS_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_EMBOSS_CARD_EX          (CEU_SERVICE_OFFSET + 4)
#define WFS_CMD_CEU_SUPPLY_REPLENISH        (CEU_SERVICE_OFFSET + 5)

/* CEU Messages */
#define WFS_SRVCE_CEU_INPUTBINTHRESHOLD     (CEU_SERVICE_OFFSET + 1)
#define WFS_SRVCE_CEU_OUTPUTBINTHRESHOLD    (CEU_SERVICE_OFFSET + 2)
#define WFS_SRVCE_CEU_RETAINBINTHRESHOLD    (CEU_SERVICE_OFFSET + 3)
#define WFS_EXECE_CEU_FIELDERROR            (CEU_SERVICE_OFFSET + 4)
#define WFS_EXECE_CEU_FIELDWARNING          (CEU_SERVICE_OFFSET + 5)
#define WFS_SRVCE_CEU_EMBOSS_FAILURE        (CEU_SERVICE_OFFSET + 6)
#define WFS_SRVCE_CEU_MEDIAREMOVED          (CEU_SERVICE_OFFSET + 7)
#define WFS_SRVCE_CEU_MEDIADETECTED         (CEU_SERVICE_OFFSET + 8)
#define WFS_SRVCE_CEU_DEVICEPOSITION        (CEU_SERVICE_OFFSET + 9)
#define WFS_SRVCE_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_DEVOPFFLINE                 WFS_STAT_DEVOPFFLINE

#ifdef __INC_XFSCEU__H
#define __INC_XFSCEU__H

#endif __cplusplus
}
#endif __cplusplus
*/

/*undef __Inc_XFSCEU__H*/
#define WFS_CEU_DEVPOWEROFF WFS_STAT_DEVPOWEROFF
#define WFS_CEU_DEVNODEVICE WFS_STAT_DEVNODEVICE
#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_MEDIAUNKNOWN (4)
#define WFS_CEU_MEDIAINHOPPER (5)
#define WFS_CEU_MEDIATOPPER (6)
#define WFS_CEU_MEDIAINHOPPER (7)
#define WFS_CEU_MEDIATOPPER (8)
#define WFS_CEU_MEDIATESTING (9)
#define WFS_CEU_MEDIARETAINED (10)
#define WFS_CEU_MEDIAREMOVED (11)
#define WFS_CEU_MEDIAPRINTED (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)
#define WFS_CEU_OUTPUTREMOVING (5)

/* values of WFSCEUSTATUS.wDevicePosition  
  WFSCEUDEVICEPOSITION.wPosition */
#define WFS_CEU_DEVICEINPOSITION (0)
#define WFS_CEU_DEVICENOTINPOSITION (1)
#define WFS_CEU_DEVICEUNKNOWN (2)
#define WFS_CEU_DEVICEUNKNOWNNOTSUPP (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_EMBOSS (0x0001)
#define WFS_CEU_PRINT (0x0002)
/* values of WFSCEUFRM_MEDIA.wBase */
#define WFS_CEU_INCH (1)
#define WFS_CEU_MM (2)
#define WFS_CEU_ROWCOLUMN (3)

/* values of WFSCEUFRM_MEDIA.fwMediaType */
#define WFS_CEU_MEDIA_ECARD (1)
#define WFS_CEU_MEDIA_PCARD (2)

/* values of WFSCEUFRM_FIELD.fwType */
#define WFS_CEU_FIELD_TEXT (1)
#define WFS_CEU_FIELD_OCR (2)

/* values of WFSCEUFRM_FIELD.fwClass */
#define WFS_CEU_CLASS_STATIC (1)
#define WFS_CEU_CLASS_OPTIONAL (2)
#define WFS_CEU_CLASS_REQUIRED (3)

/* values of WFSCEUFIELD_FAIL.wFailure */
#define WFS_CEU_FIELD_REQUIRED (1)
#define WFS_CEU_FIELD_STATIC_OVRWR (2)
#define WFS_CEU_FIELD_OVERFLOW (3)
#define WFS_CEU_FIELD_NOT_FOUND (4)
#define WFS_CEU_FIELD_NOT_READ (5)
#define WFS_CEU_FIELD_NOT_WRITE (6)
#define WFS_CEU_FIELD_HW_ERROR (7)
#define WFS_CEU_FIELD_TYPE_NOT_SUPPORTED (8)
#define WFS_CEU_CHARSET_FORM (9)

/* values of WFSCEU_EMBoss_CARD.fwChipProtocols */
#define WFS_CEU_CHIP_PROTOCOL_NOT_REQUIRED (0x0004)

/* values of WFSCEU_SUPPLY_REPLEN.fwSupplyReplen */
#define WFS_CEU_REPLEN_TONER (0x0001)
#define WFS_CEU_REPLEN_INPUTBIN (0x0002)

/* WFS_EXEE_CEU_EMBoss_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_MAGNETICStripe_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_EMBOSS_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)
```c
#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_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_EMBOSSFAILURE (-(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))

/*=================================================================* /
/* 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;
} WFSCEUCAPS, *LPWFSCEUCAPS;
```
typedef struct _wfs_ceu_form
{
    LPSTR lpszFormName;
    LPSTR lpszFields;
    WORD fwCharSupport;
    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;
    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;
    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;

/*=================================================================*/
/* 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)

#endif  /* __INC_XFSCEU__H */