CEN

WORKSHOP

AGREEMENT

CWA 16374-16

December 2011

ICS 35.240.40

English version

Extensions for Financial Services (XFS) interface specification Release 3.20 - Part 16: Card Dispenser Device Class Interface Programmer's Reference

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

Management Centre: Avenue Marnix 17, B-1000 Brussels

Table of Contents

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 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.

Revision History:

3.10	November 29, 2007	Initial release.
3.20	March 2nd, 2011	For a description of changes from version 3.10 to version 3.20 see the CRD 3.20 Migration document.

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 **WFSExecute** function, the commands to read data from various services are as similar as possible to each other in their syntax and data structures.

In general, the specific command set for a service class is defined as a superset of the specific capabilities likely to be provided by the developers of the services of that class; thus any particular device will normally support only a subset of the defined command set.

There are three cases in which a Service Provider may receive a service-specific command that it does not support:

The requested capability is defined for the class of Service Providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability is ***not*** considered to be fundamental to the service. In this case, the Service Provider returns a successful completion, but does no operation. An example would be a request from an application to turn on a control indicator on a passbook printer; the Service Provider recognizes the command, but since the passbook printer it is managing does not include that indicator, the Service Provider does no operation and returns a successful completion to the application.

The requested capability is defined for the class of Service Providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability ***is*** considered to be fundamental to the service. In this case, a WFS_ERR_UNSUPP_COMMAND error is returned to the calling application. An example would be a request from an application to a cash dispenser to dispense coins; the Service Provider recognizes the command but, since the cash dispenser it is managing dispenses only notes, returns this error.

The requested capability is ***not*** defined for the class of Service Providers by the XFS specification. In this case, a WFS_ERR_INVALID_COMMAND error is returned to the calling application.

This design allows implementation of applications that can be used with a range of services that provide differing subsets of the functionalities that are defined for their service class. Applications may use the **WFSGetInfo** and **WFSAsyncGetInfo** commands to inquire about the capabilities of the service they are about to use, and modify their behavior accordingly, or they may use functions and then deal with WFS_ERR_UNSUPP_COMMAND error returns to make decisions as to how to use the service.

2. Card Dispensers

This specification describes the functionality of the services provided by the Card Dispenser (CRD) device class under XFS, by defining the service-specific commands that can be issued, using the **WFSGetInfo, WFSAsyncGetInfo, WFSExecute** and **WFSAsyncExecute** functions.

A Card Dispenser is used to dispense a single card to a consumer from one or more bins. Most card dispensers also have the ability to retain a card to a bin.

3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer’s Reference Revision 3.20
2. XFS Identification Card Device Class Interface - Programmer's Reference, Revision 3.20

4. Info Commands

4.1 WFS_INF_CRD_STATUS

Description This command is used to request status information for the device.

Input Param None.

Output Param LPWFSCRDSTATUS lpStatus;

```
typedef struct _wfs_crd_status
     {
     WORD                fwDevice;
     WORD                fwDispenser;
     WORD                fwTransport;
     WORD                fwMedia;
     WORD                fwShutter;
     LPSTR               lpszExtra;
     DWORD               dwGuidLights[WFS_CRD_GUIDLIGHTS_SIZE];
     WORD                wDevicePosition;
     USHORT              usPowerSaveRecoveryTime;
     WORD                wAntiFraudModule;
     } WFSCRDSTATUS, *LPWFSCRDSTATUS;
```

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

Value	Meaning
WFS_CRD_DEVONLINE	The device is online (i.e. powered on and operable).
WFS_CRD_DEVOFFLINE	The device is offline (e.g. the operator has taken the device offline by turning a switch or pulling out the device).
WFS_CRD_DEVPOWEROFF	The device is powered off or physically not connected.
WFS_CRD_DEVNODEVICE	There is no device intended to be there; e.g. this type of self service machine does not contain such a device or it is internally not configured.
WFS_CRD_DEVHWERROR	The device is inoperable due to a hardware error.
WFS_CRD_DEVUSERERROR	The device is present but a person is preventing proper device operation.
WFS_CRD_DEVBUSY	The device is busy and unable to process an execute command at this time.
WFS_CRD_DEVFRAUDATTEMPT	The device is present but is inoperable because it has detected a fraud attempt.
WFS_CRD_DEVPOTENTIALFRAUD	The device has detected a potential fraud attempt and is capable of remaining in service. In this case the application should make the decision as to whether to take the device offline.

fwDispenser
Specifies the state of the card units including all retain bins as one of the following flags:

Value	Meaning
WFS_CRD_DISPCUOK	All card units present are in a good state.
WFS_CRD_DISPCUSTATE	One or more of the card units is in a low, empty or inoperative condition. Items can still be dispensed from at least one of the card units.

WFS_CRD_DISPCUSTOP	Due to a card unit failure dispensing is impossible. No items can be dispensed because all of the card units are in an empty or inoperative condition.
WFS_CRD_DISPCUUNKNOWN	Due to a hardware error or other condition, the state of the card units cannot be determined.

fwTransport
Specifies the state of the transport mechanism as one of the following values:

Value	Meaning
WFS_CRD_TPOK	The transport is in a good state.
WFS_CRD_TPINOP	The transport is inoperative due to a hardware failure or media jam.
WFS_CRD_TPUNKNOWN	Due to a hardware error or other condition, the state of the transport cannot be determined.
WFS_CRD_TPNOTSUPPORTED	The physical device has no transport or transport state reporting is not supported.

fwMedia
Specifies the state of a card that may or may not be present in the device. A card becomes media when it is moved from a dispense card unit. It will be one of the following values:

Value	Meaning
WFS_CRD_MEDIAPRESENT	Media is present in the device, but not in the exiting position and not jammed.
WFS_CRD_MEDIANOTPRESENT	Media is not present in the device and not at the exiting position.
WFS_CRD_MEDIAJAMMED	Media is jammed in the device.
WFS_CRD_MEDIANOTSUPP	Capability to report media position is not supported by the device.
WFS_CRD_MEDIAUNKNOWN	The media state cannot be determined with the device in its current state.
WFS_CRD_MEDIAEXITING	Media is at the exit slot of the card dispenser unit.

fwShutter
Specifies the state of the shutter as one of the following flags:

Value	Meaning
WFS_CRD_SHTCLOSED	The shutter is closed.
WFS_CRD_SHTOPEN	The shutter is opened.
WFS_CRD_SHTJAMMED	The shutter is jammed.
WFS_CRD_SHTUNKNOWN	Due to a hardware error or other condition, the state of the shutter cannot be determined.
WFS_CRD_SHTNOTSUPPORTED	The physical device has no shutter or shutter state reporting is not supported.

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.

dwGuidLights [...]
Specifies the state of the guidance light indicators. A number of guidance light types are defined below. Vendor specific guidance lights are defined starting from the end of the array. The maximum guidance light index is WFS_CRD_GUIDLIGHTS_MAX.

Specifies the state of the guidance light indicator as WFS_CRD_GUIDANCE_NOT_AVAILABLE, WFS_CRD_GUIDANCE_OFF or a combination of the following flags consisting of one type B, and optionally one type C.

Value	Meaning	Type
WFS_CRD_GUIDANCE_NOT_AVAILABLE	The status is not available.	A
WFS_CRD_GUIDANCE_OFF	The light is turned off.	A
WFS_CRD_GUIDANCE_SLOW_FLASH	The light is blinking slowly.	B
WFS_CRD_GUIDANCE_MEDIUM_FLASH	The light is blinking medium frequency.	B
WFS_CRD_GUIDANCE_QUICK_FLASH	The light is blinking quickly.	B
WFS_CRD_GUIDANCE_CONTINUOUS	The light is turned on continuous (steady).	B
WFS_CRD_GUIDANCE_RED	The light is red.	C
WFS_CRD_GUIDANCE_GREEN	The light is green.	C
WFS_CRD_GUIDANCE_YELLOW	The light is yellow.	C
WFS_CRD_GUIDANCE_BLUE	The light is blue.	C
WFS_CRD_GUIDANCE_CYAN	The light is cyan.	C
WFS_CRD_GUIDANCE_MAGENTA	The light is magenta.	C
WFS_CRD_GUIDANCE_WHITE	The light is white.	C

dwGuidLights [WFS_CRD_GUIDANCE_CARDDISP]
Specifies the state of the guidance light indicator on the card dispensing unit.

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_CRD_DEVICENOTINPOSITION, *fwDevice* can have any of the values defined above (including WFS_CRD_DEVONLINE or WFS_CRD_DEVOFFLINE). If the device is not in its normal operating position (i.e. WFS_CRD_DEVICEINPOSITION) then media may not be presented through the normal customer interface. This value is one of the following values:

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

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

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

Value	Meaning
WFS_CRD_AFMNOTSUPP	No anti-fraud module is available.
WFS_CRD_AFMOK	Anti-fraud module is in a good state and no foreign device is detected.
WFS_CRD_AFMINOP	Anti-fraud module is inoperable.
WFS_CRD_AFMDEVICEDETECTED	Anti-fraud module detected the presence of a foreign device.
WFS_CRD_AFMUNKNOWN	The state of the anti-fraud module cannot be determined.

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

Comments Applications which require or expect specific information to be present in the *lpszExtra* parameter may not be device or vendor-independent. If the CRD device is part of a compound device with an IDC device then a dispense to transport can allow the IDC interface to be used to read/write from the card.

In the case where communications with the device has been lost, the *fwDevice* field will report WFS_CRD_DEVPOWEROFF when the device has been removed or WFS_CRD_DEVHWERROR if the communications are unexpectedly lost. All other fields will report their status as unknown.

4.2 WFS_INF_CRD_CAPABILITIES

Description This command is used to request device capability information.

Input Param None.

Output Param LPWFSCRDCAPS lpCaps;

```
typedef struct _wfs_crd_caps
     {
     WORD                wClass;
     BOOL                bCompound;
     WORD                fwPowerOnOption;
     WORD                fwPowerOffOption;
     BOOL                bCardTakenSensor;
     WORD                fwDispenseTo;
     LPSTR               lpszExtra;
     DWORD               dwGuidLights[WFS_CRD_GUIDLIGHTS_SIZE];
     BOOL                bPowerSaveControl;
     BOOL                bAntiFraudModule;
     } WFSCRDCAPS, *LPWFSCRDCAPS;
```

wClass
Specifies the logical service class as WFS_SERVICE_CLASS_CRD.

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

fwPowerOnOption
Specifies the power-on capabilities of the device hardware, as one of the following flags; applicable only to motor driven ID card units.

Value	Meaning
WFS_CRD_NOACTION	No power on actions are supported by the device.
WFS_CRD_EJECT	The card will be ejected on power-on (or off, see *fwPowerOffOption* below).
WFS_CRD_RETAIN	The card will be retained on power-on (off).
WFS_CRD_EJECTTHENRETAIN	The card will be ejected for a specified time after power-on then retained if not taken. The time for which the card is ejected is vendor dependent.

fwPowerOffOption
Specifies the power-off capabilities of the device hardware, as one of the flags specified for *fwPowerOnOption*.

bCardTakenSensor
Specifies whether or not the card dispenser has the ability to detect when a card is taken from the exit slot by a user. TRUE means a sensor exists and the “card taken” condition can be detected. In this case a WFS_SRVE_CRD MEDIATAKEN event will be generated when the card is removed. If set to FALSE then no event will be generated.

fwDispenseTo
Specifies where a card will be dispensed to as a combination of the following flags:

Value	Meaning
WFS_CRD_DISPTO_CONSUMER	A dispensed card can be delivered to the exit slot for the consumer to take.
WFS_CRD_DISPTO_TRANSPORT	A dispensed card can be delivered into the transport mechanism. The application must use WFS_CMD_CRD_EJECT_CARD to deliver the card to the consumer.

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.

dwGuidLights [...]
Specifies which guidance lights are available. A number of guidance light types are defined below. Vendor specific guidance lights are defined starting from the end of the array. The maximum guidance light index is WFS_CRD_GUIDLIGHTS_MAX.

The elements of this array are specified as a combination of the following flags and indicate all of the possible flash rates (type B) and colors (type C) that the guidance light indicator is capable of handling. If the guidance light indicator only supports one color then no value of type C is returned. A value of WFS_CRD_GUIDANCE_NOT_AVAILABLE indicates that the device has no guidance light indicator or the device controls the light directly with no application control possible.

Value	Meaning	Type
WFS_CRD_GUIDANCE_NOT_AVAILABLE	There is no guidance light control available at this position.	A
WFS_CRD_GUIDANCE_OFF	The light can be off.	B
WFS_CRD_GUIDANCE_SLOW_FLASH	The light can blink slowly.	B
WFS_CRD_GUIDANCE_MEDIUM_FLASH	The light can blink medium frequency.	B
WFS_CRD_GUIDANCE_QUICK_FLASH	The light can blink quickly.	B
WFS_CRD_GUIDANCE_CONTINUOUS	The light can be continuous (steady).	B
WFS_CRD_GUIDANCE_RED	The light can be red.	C
WFS_CRD_GUIDANCE_GREEN	The light can be green.	C
WFS_CRD_GUIDANCE_YELLOW	The light can be yellow.	C
WFS_CRD_GUIDANCE_BLUE	The light can be blue.	C
WFS_CRD_GUIDANCE_CYAN	The light can be cyan.	C
WFS_CRD_GUIDANCE_MAGENTA	The light can be magenta.	C
WFS_CRD_GUIDANCE_WHITE	The light can be white.	C

dwGuidLights [WFS_CRD_GUIDANCE_CARDDISP
Specifies whether the guidance light indicator on the card unit is available.

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

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. If the CRD device is part of a compound device with an IDC device then a dispense to transport can allow the IDC interface to be used to read/write from the card.

4.3 WFS_INF_CRD_CARD_UNIT_INFO

Description This command is used to obtain information regarding the status and contents of the card units in the CRD.

Counts
The values of the following fields of the WFSCRDCARDUNIT
ulCount
ulRetainCount
are persistent software counts and therefore may not represent the actual number of items in the card unit.

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

Threshold Events
The threshold event WFS_USRE_CRD_CARDUNITTHRESHOLD can be triggered either by hardware sensors in the device or by the *ulCount* reaching the *ulThreshold* value.

The application can check if the device has this capability by querying the *bHardwareSensor* field of the card unit structure.

Input Param None.

Output Param LPWFSCRDCUINFO lpCardUnitInfo;

```
typedef struct _wfs_crd_cu_info
     {
     USHORT               usCount;
     LPWFSCRDCARDUNIT     *lppList;
     } WFSCRDCUINFO, *LPWFSCRDCUINFO;
```

usCount
Specifies the number of card unit structures returned.

lppList
Pointer to an array of pointers to WFSCRDCARDUNIT structures:

```
typedef struct _wfs_crd_cardunit
     {
     USHORT               usNumber;
     LPSTR                lpszCardName;
     USHORT               usType;
     ULONG                ulInitialCount;
     ULONG                ulCount;
     ULONG                ulRetainCount;
     ULONG                ulThreshold;
     USHORT               usStatus;
     BOOL                 bHardwareSensor;
     } WFSCRDCARDUNIT, *LPWFSCRDCARDUNIT;
```

usNumber
Index number of the card unit structure. Each structure has a unique logical number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.

lpszCardName
An identifier which is used to identify the type of cards in the card unit.

usType
Type of card unit as one of the following values:

Value	Meaning
WFS_CRD_SUPPLYBIN	The card unit is a supply card unit.
WFS_CRD_RETAINBIN	The card unit is a retain card unit.

ulInitialCount
Initial number of items contained in the card unit. This value is persistent.

ulCount
The number of items inside the card unit plus any items from the card units not yet presented to the customer. This count is decremented when the items are either presented to the customer or retained. This count is incremented for a retain bin after a retain operation.

If this value reaches zero it will not decrement further but will remain at zero. This value is persistent.

ulRetainCount
The number of items from this card unit which are in the retain bin. This field is always zero for a retain bin. This value is persistent.

ulThreshold
When *ulCount* reaches this value the WFS_USRE_CRD_CARDUNITTHRESHOLD threshold event will be generated. A WFS_CRD_STATCUHIGH threshold will be sent for WFS_CRD_RETAINBIN or WFS_CRD_STATCULOW for a WFS_CRD_SUPPLYBIN. If this value is non-zero then hardware sensors in the device do not trigger threshold events.

usStatus
Supplies the status of the card unit as one of the following values:

Value	Meaning
WFS_CRD_STATCUOK	The card supply or retain unit is in a good state.
WFS_CRD_STATCULOW	The card supply unit is almost empty.
WFS_CRD_STATCUEMPTY	The card supply unit is empty.
WFS_CRD_STATCUINOP	The card supply or retain unit is inoperative.
WFS_CRD_STATCUMISSING	The card supply or retain unit is missing.
WFS_CRD_STATCUHIGH	The retain card unit is almost full.
WFS_CRD_STATCUFULL	The retain card unit is full.
WFS_CRD_STATCUUNKNOWN	The status of the card unit cannot be determined.

bHardwareSensor
Specifies whether or not threshold events can be generated based on hardware sensors in the device. This applies to WFS_CRD_STATCULOW and WFS_CRD_STATCUHIGH thresholds only. If this value is TRUE then threshold events may be generated based on hardware sensors as opposed to counts. If *ulThreshold* is non zero then hardware triggers are ignored and software trigger/counters are used. A WFS_CRD_STATCUHIGH threshold will be sent for a retain bin or WFS_CRD_STATCULOW for a card supply unit. This field is read only.

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

Comments None.

5. Execute Commands

5.1 WFS_CMD_CRD_DISPENSE_CARD

Description This command will attempt to move a card from the internal supply to a dispensable position. If the card is only dispensed to the transport then the command WFS_CMD_CRD_EJECT_CARD should be used to get the card in a position that the consumer can take it.

If the CRD Service Provider is a compound device with the IDC class, then when the card has been successfully dispensed and is in the transport it can be treated like any other inserted card on the IDC interface. For example, if the device has read/write capabilities the card can be written to and read from using the IDC commands.

Input Param LPWFSCRDDISPENSE lpDispense;

```
typedef struct _wfs_crd_dispense
     {
     USHORT               usNumber;
     BOOL                 bPresent;
     } WFSCRDDISPENSE, *LPWFSCRDDISPENSE;
```

usNumber
The number of the card unit from which the card should be dispensed. The number of the card unit is the *usNumber* returned from WFS_INF_CRD_UNIT_INFO.

bPresent
If this field is set to TRUE then the items will be moved to the exit slot, if it is FALSE the items will be moved to the transport. The *bPresent* flag will be ignored if the device cannot dispense to the transport.

Output Param None.

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

Value	Meaning
WFS_ERR_CRD_MEDIAJAM	The card is jammed in the transport.
WFS_ERR_CRD_DEVICE_OCCUPIED	There is already a card in the dispensing device. A second card cannot be dispensed.
WFS_ERR_CRD_SHUTTERFAIL	The open of the shutter failed due to manipulation or hardware error.
WFS_ERR_CRD_CARDUNITERROR	There is a problem with a card unit. The WFS_EXEE_CRD_CARDUNITERROR execute event is posted with the details.
WFS_ERR_CRD_MEDIARETAINED	The dispense operation failed the card has been retained and the device is clear.

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

Value	Meaning
WFS_EXEE_CRD_CARDUNITERROR	A card unit caused an error during a dispense operation.
WFS_USRE_CRD_CARDUNITTHRESHOLD	A card unit has reached a threshold.
WFS_SRVE_CRD_MEDIAREMOVED	The card has been taken by the user.

Comments None.

5.2 WFS_CMD_CRD_EJECT_CARD

Description This command only needs to be used if the card is not delivered all the way to the exit slot during a WFS_CMD_CRD_DISPENSE_CARD operation. An example of this would be for a compound device where the card is dispensed only to the transport for reading/writing to the magnetic stripe. After the card is read from and/or written to, the card can then be ejected to the exit for removal by the consumer.

Input Param None.

Output Param None.

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

Value	Meaning
WFS_ERR_CRD_MEDIAJAM	The card is jammed.
WFS_ERR_CRD_SHUTTERFAIL	The open of the shutter failed due to manipulation or hardware error.
WFS_ERR_CRD_NOMEDIA	No card is present.
WFS_ERR_CRD_MEDIARETAINED	The card has been retained during attempts to eject it. The device is clear and can be used.

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

Value	Meaning
WFS_SRVE_CRD_MEDIAREMOVED	The card has been taken by the user.
WFS_USRE_CRD_CARDUNITTHRESHOLD	A card unit has reached a threshold.

Comments None.

5.3 WFS_CMD_CRD_RETAIN_CARD

Description The card is removed from its present position and stored in a retain bin. The card dispensing unit sends an event if the storage capacity of the retain bin is reached. If the storage capacity has already been reached, and the command cannot be executed then the card remains in its present position and a WFS_ERR_CRD_RETAINBINFULL error is returned.

Input Param LPWFSCRDRETAINCARD lpRetainCard;

```
typedef struct _wfs_crd_retain_card
     {
     USHORT               usNumber;
     } WFSCRDRETAINCARD, *LPWFSCRDRETAINCARD;
```

usNumber
The number of the retain bin that the card is to be retracted to. This corresponds to the *usNumber* returned by the WFS_INF_CRD_CARD_UNIT_INFO command and must represent a retain bin.

Output Param None.

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

Value	Meaning
WFS_ERR_CRD_MEDIAJAM	The card is jammed.
WFS_ERR_CRD_NOMEDIA	There is no card to retain.
WFS_ERR_CRD_RETAINBINFULL	The retain bin is full; no more cards can be retained. The current card is still in the device.
WFS_ERR_CRD_CARDUNITERROR	A card unit caused an error.
WFS_ERR_CRD_INVALIDRETAINBIN	The retain bin specified in the *usNumber* input parameter is invalid.

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

Value	Meaning
WFS_USRE_CRD_CARDUNITTHRESHOLD	A card unit has reached a threshold.
WFS_SRVE_CRD_MEDIAREMOVED	The card has been taken by the user.

Comments If a retain request is received by a device with no retain capability, the WFS_ERR_UNSUPP_COMMAND error is returned.

5.4 WFS_CMD_CRD_RESET

Description This command is used by the application to perform a hardware reset which will attempt to return the CRD device to a known good state. This command does not over-ride a lock obtained by another application or service handle.

Input Param LPWFSCRDRESET lpResetIn;

```
typedef struct _wfs_crd_reset
     {
     USHORT               usAction;
     } WFSCRDRESET, *LPWFSCRDRESET;
```

usAction
Specifies the action to be performed on any card found within the device as one of the following values:

Value	Meaning
WFS_CRD_EJECT	Eject any card found.
WFS_CRD_RETAIN	Retain any card found.
WFS_CRD_NOACTION	No action should be performed on any card found.

If the application does not wish to specify an action it can set *lpResetIn* to NULL. In this case the Service Provider will determine where to move the card.

Output Param None.

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

Value	Meaning
WFS_ERR_CRD_MEDIAJAM	A card is jammed. Operator intervention is required.
WFS_ERR_CRD_SHUTTERFAIL	The device is unable to open and close its shutter.

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

Value	Meaning
WFS_SRVE_CRD_MEDIADETECTED	This event is generated when media is detected during a reset.
WFS_USRE_CRD_CARDUNITTHRESHOLD	A card unit has reached a threshold.
WFS_SRVE_CRD_MEDIAREMOVED	The card has been taken by the user.

Comments None.

5.5 WFS_CMD_CRD_SET_CARD_UNIT_INFO

Description This command is used to adjust information regarding the contents of the card units present in the CRD. Some fields may be ignored by the Service Provider if the information can be obtained from the device. In some cases the fields that can be set is dependent on Service Provider configuration.

The following fields cannot be changed using this command.

usNumber
usType
usStatus
bHardwareSensor

This command generates the service event WFS_SRVE_CRD_CARDUNITINFOCHANGED to inform applications that the information for a card unit has been changed.

Input Param LPWFSCRDCUINFO lpCUInfo;

The WFSCRDCUINFO structure is specified in the documentation of the WFS_INF_CRD_CARD_UNIT_INFO command. This structure contains all of the card units reported by the WFSCRDCUINFO command.

Output Param None.

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

Value	Meaning
WFS_ERR_CRD_INVALIDCARDUNIT	Invalid card unit.

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

Value	Meaning
WFS_USRE_CRD_CARDUNITTHRESHOLD	A card unit has reached a threshold or a threshold has been cleared.
WFS_SRVE_CRD_CARDUNITINFOCHANGED	A card unit was updated as a result of this command.

Comments None.

5.6 WFS_CMD_CRD_SET_GUIDANCE_LIGHT

Description This command is used to set the status of the CRD guidance lights. This includes defining the flash rate and the color. When an application tries to use a color that is not supported then the Service Provider will return the generic error WFS_ERR_UNSUPP_DATA.

Input Param LPWFSCRDSETGUIDLIGHT lpSetGuidLight;

```
typedef struct _wfs_crd_set_guidlight
     {
     WORD                wGuidLight;
     DWORD               dwCommand;
     } WFSCRDSETGUIDLIGHT, *LPWFSCRDSETGUIDLIGHT;
```

wGuidLight
Specifies the index of the guidance light to set as one of the values defined within the capabilities section.

dwCommand
Specifies the state of the guidance light indicator as WFS_CRD_GUIDANCE_OFF or a combination of the following flags consisting of one type B, and optionally one type C. If no value of type C is specified then the default color is used. The Service Provider determines which color is used as the default color.

Value	Meaning	Type
WFS_CRD_GUIDANCE_OFF	The light indicator is turned off.	A
WFS_CRD_GUIDANCE_SLOW_FLASH	The light indicator is set to flash slowly.	B
WFS_CRD_GUIDANCE_MEDIUM_FLASH	The light indicator is set to flash medium frequency.	B
WFS_CRD_GUIDANCE_QUICK_FLASH	The light indicator is set to flash quickly.	B
WFS_CRD_GUIDANCE_CONTINUOUS	The light indicator is turned on continuously (steady).	B
WFS_CRD_GUIDANCE_RED	The light indicator color is set to red.	C
WFS_CRD_GUIDANCE_GREEN	The light indicator color is set to green.	C
WFS_CRD_GUIDANCE_YELLOW	The light indicator color is set to yellow.	C
WFS_CRD_GUIDANCE_BLUE	The light indicator color is set to blue.	C
WFS_CRD_GUIDANCE_CYAN	The light indicator color is set to cyan.	C
WFS_CRD_GUIDANCE_MAGENTA	The light indicator color is set to magenta.	C
WFS_CRD_GUIDANCE_WHITE	The light indicator color is set to white.	C

Output Param None.

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

Value	Meaning
WFS_ERR_CRD_INVALID_PORT	An attempt to set a guidance light to a new value was invalid because the guidance light does not exist.

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

Comments Guidance light support was added into the CRD primarily to support guidance lights for workstations where more than one instance of a CRD is present. The original CRD guidance light mechanism was not able to manage guidance lights for workstations with multiple CRDs. This command can also be used to set the status of the CRD guidance lights when only one instance of a CRD is present.

The slow and medium flash rates must not be greater than 2.0 Hz. It should be noted that in order to comply with American Disabilities Act guidelines only a slow or medium flash rate must be used.

5.7 WFS_CMD_CRD_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 LPWFSCRDPOWERSAVECONTROL lpPowerSaveControl;

```
typedef struct _wfs_crd_power_save_control
     {
     USHORT               usMaxPowerSaveRecoveryTime;
     } WFSCRDPOWERSAVECONTROL, *LPWFSCRDPOWERSAVECONTROL;
```

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

Output Param None.

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

Value	Meaning
WFS_ERR_CRD_POWERSAVETOOSHORT	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.
WFS_ERR_CRD_POWERSAVEMEDIAPRESENT	The power saving mode has not been activated because media is present inside the device.

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

Value	Meaning
WFS_SRVE_CRD_POWER_SAVE_CHANGE	The power save recovery time has changed.

Comments None.

6. Events

6.1 WFS_SRVE_CRD_MEDIAREMOVED

Description	This event is sent when the media is taken from the exit slot.
Event Param	None.
Comments	This event occurs after the completion of a function that ejects the media, it is not an execute event.

6.2 WFS_SRVE_CRD_MEDIADETECTED

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

Event Param LPWFSCRDMEDIADETECTED lpMediaDetected;

```
typedef struct _wfs_crd_media_detected
     {
     WORD                wPosition;
     USHORT              usNumber;
     } WFSCRDMEDIADETECTED, *LPWFSCRDMEDIADETECTED;
```

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

Value	Meaning
WFS_CRD_MEDIARETAINED	The media was retained during the reset operation.
WFS_CRD_MEDIAPRESENT	The media is present somewhere in the transport.
WFS_CRD_MEDIAEXITING	The media is in the exit slot.
WFS_CRD_MEDIAJAMMED	The media is jammed in the device.
WFS_CRD_MEDIAUNKNOWN	The media is in an unknown position.

usNumber
Number of the retain bin the media was retained to. This number has to be between one and the number of bins supported by this device. It is only relevant if *wPosition* equals WFS_CRD_MEDIARETAINED.

Comments None.

6.3 WFS_USRE_CRD_CARDUNITTHRESHOLD

Description	This user event is generated when a threshold condition has occurred in one of the card units.
Event Param	LPWFSCRDCARDUNIT lpCardUnit; *lpCardUnit* Pointer to a WFSCRDCARDUNIT structure describing the card unit on which the threshold condition occurred. See *lpCardUnit->usStatus* for the current status. For a description of the WFSCRDCARDUNIT structure see the definition of the WFS_INF_CRD_CARD_UNIT_INFO command.
Comments	None.

6.4 WFS_SRVE_CRD_CARDUNITINFOCHANGED

Description	This service event is generated when information about a card unit has changed. This event will also be posted on successful completion of the following commands: WFS_CMD_CRD_SET_CARD_UNIT_INFO
Event Param	LPWFSCRDCARDUNIT lpCardUnit; *lpCardUnit* Pointer to the changed card unit structure. For a description of the WFSCRDCARDUNIT structure see the definition of the WFS_INF_CRD_CARD_UNIT_INFO command.
Comments	None.

6.5 WFS_EXEE_CRD_CARDUNITERROR

Description This execute event is generated if there is a problem with a card unit during a dispense operation.

Event Param LPWFSCRDCUERROR lpCardUnitError;

```
typedef struct _wfs_crd_cu_error
     {
     WORD                 wFailure;
     LPWFSCRDCARDUNIT     lpCardUnit;
     } WFSCRDCUERROR, *LPWFSCRDCUERROR;
```

wFailure
Specifies the kind of failure that occurred in the card unit. Values are:

Value	Meaning
WFS_CRD_CARDUNITEMPTY	Specified card unit is empty.
WFS_CRD_CARDUNITERROR	Specified card unit has malfunctioned.
WFS_CRD_CARDUNITINVALID	Specified card unit ID is invalid.

lpCardUnit
Pointer to the card unit structure that caused the problem. The WFSCRDCARDUNIT structure is defined in the documentation of the WFS_INF_CRD_CARD_UNIT_INFO command. It is possible that this pointer may be NULL if the *wFailure* field is WFS_CRD_CARDUNITINVALID.

Comments None.

6.6 WFS_SRVE_CRD_DEVICEPOSITION

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

Event Param LPWFSCRDDEVICEPOSITION lpDevicePosition;

```
typedef struct _wfs_crd_device_position
     {
     WORD                 wPosition;
     } WFSCRDDEVICEPOSITION, *LPWFSCRDDEVICEPOSITION;
```

wPosition
Position of the device as one of the following values:

Value	Meaning
WFS_CRD_DEVICEINPOSITION	The device is in its normal operating position.
WFS_CRD_DEVICENOTINPOSITION	The device has been removed from its normal operating position.
WFS_CRD_DEVICEPOSUNKNOWN	The position of the device cannot be determined.

Comments None.

6.7 WFS_SRVE_CRD_POWER_SAVE_CHANGE

Description This service event specifies that the power save recovery time has changed.

Event Param LPWFSCRDPOWERSAVECHANGE lpPowerSaveChange;

```
typedef struct _wfs_crd_power_save_change
     {
     USHORT               usPowerSaveRecoveryTime;
     } WFSCRDPOWERSAVECHANGE, *LPWFSCRDPOWERSAVECHANGE;
```

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.

7. C-Header File

```
/******************************************************************************
*                                                                             *
* xfscrd.h    XFS - Card Dispenser (CRD) definitions                          *
*                                                                             *
*             Version 3.20  (March 02 2011)                                   *
*                                                                             *
******************************************************************************/

#ifndef __INC_XFSCRD__H
#define __INC_XFSCRD__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

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

/* values of WFSCRDCAPS.wClass */

#define     WFS_SERVICE_CLASS_CRD               (14)
#define     WFS_SERVICE_CLASS_VERSION_CRD       (0x1403) /* Version 3.20 */
#define     WFS_SERVICE_CLASS_NAME_CRD          "CRD"

#define     CRD_SERVICE_OFFSET                  (WFS_SERVICE_CLASS_CRD * 100)

/* CRD Info Commands */

#define     WFS_INF_CRD_STATUS                  (CRD_SERVICE_OFFSET + 1)
#define     WFS_INF_CRD_CAPABILITIES            (CRD_SERVICE_OFFSET + 2)
#define     WFS_INF_CRD_CARD_UNIT_INFO          (CRD_SERVICE_OFFSET + 3)

/* CRD Execute Commands */

#define     WFS_CMD_CRD_DISPENSE_CARD           (CRD_SERVICE_OFFSET + 1)
#define     WFS_CMD_CRD_EJECT_CARD              (CRD_SERVICE_OFFSET + 2)
#define     WFS_CMD_CRD_RETAIN_CARD             (CRD_SERVICE_OFFSET + 3)
#define     WFS_CMD_CRD_RESET                   (CRD_SERVICE_OFFSET + 4)
#define     WFS_CMD_CRD_SET_GUIDANCE_LIGHT      (CRD_SERVICE_OFFSET + 5)
#define     WFS_CMD_CRD_SET_CARD_UNIT_INFO      (CRD_SERVICE_OFFSET + 6)
#define     WFS_CMD_CRD_POWER_SAVE_CONTROL      (CRD_SERVICE_OFFSET + 7)

/* CRD Events  */

#define     WFS_SRVE_CRD_MEDIAREMOVED           (CRD_SERVICE_OFFSET + 1)
#define     WFS_SRVE_CRD_CARDUNITINFOCHANGED    (CRD_SERVICE_OFFSET + 2)
#define     WFS_SRVE_CRD_MEDIADETECTED          (CRD_SERVICE_OFFSET + 3)
#define     WFS_USRE_CRD_CARDUNITTHRESHOLD      (CRD_SERVICE_OFFSET + 4)
#define     WFS_EXEE_CRD_CARDUNITERROR          (CRD_SERVICE_OFFSET + 5)
#define     WFS_SRVE_CRD_DEVICEPOSITION         (CRD_SERVICE_OFFSET + 6)
#define     WFS_SRVE_CRD_POWER_SAVE_CHANGE      (CRD_SERVICE_OFFSET + 7)

/* values of WFSCRDSTATUS.fwDevice */

#define     WFS_CRD_DEVONLINE                   WFS_STAT_DEVONLINE
#define     WFS_CRD_DEVOFFLINE                  WFS_STAT_DEVOFFLINE
#define     WFS_CRD_DEVPOWEROFF                 WFS_STAT_DEVPOWEROFF
#define     WFS_CRD_DEVNODEVICE                 WFS_STAT_DEVNODEVICE
#define     WFS_CRD_DEVHWERROR                  WFS_STAT_DEVHWERROR
#define     WFS_CRD_DEVUSERERROR                WFS_STAT_DEVUSERERROR
#define     WFS_CRD_DEVBUSY                     WFS_STAT_DEVBUSY
#define     WFS_CRD_DEVFRAUDATTEMPT             WFS_STAT_DEVFRAUDATTEMPT
#define     WFS_CRD_DEVPOTENTIALFRAUD           WFS_STAT_DEVPOTENTIALFRAUD

/* values of WFSCRDSTATUS.fwDispenser */
```

```
#define     WFS_CRD_DISPCUOK                    (0)
#define     WFS_CRD_DISPCUSTATE                 (1)
#define     WFS_CRD_DISPCUSTOP                  (2)
#define     WFS_CRD_DISPCUUNKNOWN               (3)

/* values of WFSCRDSTATUS.fwMedia,
             WFSCRDRETAINCARD.fwPosition, and
             WFSCRDMEDIADETECTED.wPosition */

#define     WFS_CRD_MEDIAPRESENT                (1)
#define     WFS_CRD_MEDIANOTPRESENT             (2)
#define     WFS_CRD_MEDIAJAMMED                 (3)
#define     WFS_CRD_MEDIANOTSUPP                (4)
#define     WFS_CRD_MEDIAUNKNOWN                (5)
#define     WFS_CRD_MEDIAEXITING                (6)
#define     WFS_CRD_MEDIARETAINED               (7)

/* values of WFSCRDSTATUS.fwTransport */

#define     WFS_CRD_TPOK                        (0)
#define     WFS_CRD_TPINOP                      (1)
#define     WFS_CRD_TPUNKNOWN                   (2)
#define     WFS_CRD_TPNOTSUPPORTED              (3)

/* Size and max index of dwGuidLights array */

#define     WFS_CRD_GUIDLIGHTS_SIZE             (32)
#define     WFS_CRD_GUIDLIGHTS_MAX              (WFS_CRD_GUIDLIGHTS_SIZE - 1)

/* Indices of WFSCRDSTATUS.dwGuidLights [...]
              WFSCRDCAPS.dwGuidLights [...] */

#define     WFS_CRD_GUIDANCE_CARDDISP           (0)

/* Values of WFSCRDSTATUS.dwGuidLights [...]
             WFSCRDCAPS.dwGuidLights [...] */

#define     WFS_CRD_GUIDANCE_NOT_AVAILABLE      (0x00000000)
#define     WFS_CRD_GUIDANCE_OFF                (0x00000001)
#define     WFS_CRD_GUIDANCE_SLOW_FLASH         (0x00000004)
#define     WFS_CRD_GUIDANCE_MEDIUM_FLASH       (0x00000008)
#define     WFS_CRD_GUIDANCE_QUICK_FLASH        (0x00000010)
#define     WFS_CRD_GUIDANCE_CONTINUOUS         (0x00000080)
#define     WFS_CRD_GUIDANCE_RED                (0x00000100)
#define     WFS_CRD_GUIDANCE_GREEN              (0x00000200)
#define     WFS_CRD_GUIDANCE_YELLOW             (0x00000400)
#define     WFS_CRD_GUIDANCE_BLUE               (0x00000800)
#define     WFS_CRD_GUIDANCE_CYAN               (0x00001000)
#define     WFS_CRD_GUIDANCE_MAGENTA            (0x00002000)
#define     WFS_CRD_GUIDANCE_WHITE              (0x00004000)

/* values of WFSCRDSTATUS.wDevicePosition
             WFSCRDDEVICEPOSITION.wPosition */

#define     WFS_CRD_DEVICEINPOSITION            (0)
#define     WFS_CRD_DEVICENOTINPOSITION         (1)
#define     WFS_CRD_DEVICEPOSUNKNOWN            (2)
#define     WFS_CRD_DEVICEPOSNOTSUPP            (3)

/*values of WFSCRDCAPS.fwDispenseTo */

#define     WFS_CRD_DISPTO_CONSUMER             (0x0001)
#define     WFS_CRD_DISPTO_TRANSPORT            (0x0002)

/*values of WFSCRDCARDUNIT.usStatus */

#define    WFS_CRD_STATCUOK                     (0)
#define    WFS_CRD_STATCULOW                    (1)
#define    WFS_CRD_STATCUEMPTY                  (2)
```

```
#define    WFS_CRD_STATCUINOP                  (3)
#define    WFS_CRD_STATCUMISSING               (4)
#define    WFS_CRD_STATCUHIGH                  (5)
#define    WFS_CRD_STATCUFULL                  (6)
#define    WFS_CRD_STATCUUNKNOWN               (7)

/*values of WFSCRDCARDUNIT.usType */

#define    WFS_CRD_SUPPLYBIN                   (1)
#define    WFS_CRD_RETAINBIN                   (2)

/* values of WFSCRDSTATUS.fwShutter */

#define     WFS_CRD_SHTCLOSED                  (0)
#define     WFS_CRD_SHTOPEN                    (1)
#define     WFS_CRD_SHTJAMMED                  (2)
#define     WFS_CRD_SHTUNKNOWN                 (3)
#define     WFS_CRD_SHTNOTSUPPORTED            (4)

/* values of WFSCRDCAPS.fwPowerOnOption,
             WFSCRDCAPS.fwPowerOffOption,
             WFSCRDRESET.usAction  */

#define     WFS_CRD_NOACTION                   (1)
#define     WFS_CRD_EJECT                      (2)
#define     WFS_CRD_RETAIN                     (3)
#define     WFS_CRD_EJECTTHENRETAIN            (4)

/*values of WFSCRDCUERROR.wFailure */

#define WFS_CRD_CARDUNITEMPTY                  (1)
#define WFS_CRD_CARDUNITERROR                  (2)
#define WFS_CRD_CARDUNITINVALID                (3)

/* values of WFSCRDSTATUS.wAntiFraudModule */

#define     WFS_CRD_AFMNOTSUPP                 (0)
#define     WFS_CRD_AFMOK                      (1)
#define     WFS_CRD_AFMINOP                    (2)
#define     WFS_CRD_AFMDEVICEDETECTED          (3)
#define     WFS_CRD_AFMUNKNOWN                 (4)

/* XFS CRD Errors */

#define WFS_ERR_CRD_MEDIAJAM                   (-(CRD_SERVICE_OFFSET + 0))
#define WFS_ERR_CRD_NOMEDIA                    (-(CRD_SERVICE_OFFSET + 1))
#define WFS_ERR_CRD_MEDIARETAINED              (-(CRD_SERVICE_OFFSET + 2))
#define WFS_ERR_CRD_RETAINBINFULL              (-(CRD_SERVICE_OFFSET + 3))
#define WFS_ERR_CRD_SHUTTERFAIL                (-(CRD_SERVICE_OFFSET + 4))
#define WFS_ERR_CRD_DEVICE_OCCUPIED            (-(CRD_SERVICE_OFFSET + 5))
#define WFS_ERR_CRD_CARDUNITERROR              (-(CRD_SERVICE_OFFSET + 6))
#define WFS_ERR_CRD_INVALIDCARDUNIT            (-(CRD_SERVICE_OFFSET + 7))
#define WFS_ERR_CRD_INVALID_PORT               (-(CRD_SERVICE_OFFSET + 8))
#define WFS_ERR_CRD_INVALIDRETAINBIN           (-(CRD_SERVICE_OFFSET + 9))
#define WFS_ERR_CRD_POWERSAVETOOSHORT          (-(CRD_SERVICE_OFFSET + 10))
#define WFS_ERR_CRD_POWERSAVEMEDIAPRESENT      (-(CRD_SERVICE_OFFSET + 11))

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

typedef struct _wfs_crd_status
{
    WORD             fwDevice;
    WORD             fwDispenser;
    WORD             fwTransport;
    WORD             fwMedia;
    WORD             fwShutter;
    LPSTR            lpszExtra;
    DWORD            dwGuidLights[WFS_CRD_GUIDLIGHTS_SIZE];
    WORD             wDevicePosition;
    USHORT           usPowerSaveRecoveryTime;
    WORD             wAntiFraudModule;
} WFSCRDSTATUS, *LPWFSCRDSTATUS;

typedef struct _wfs_crd_caps
{
    WORD             wClass;
    BOOL             bCompound;
    WORD             fwPowerOnOption;
    WORD             fwPowerOffOption;
    BOOL             bCardTakenSensor;
    WORD             fwDispenseTo;
    LPSTR            lpszExtra;
    DWORD            dwGuidLights[WFS_CRD_GUIDLIGHTS_SIZE];
    BOOL             bPowerSaveControl;
    BOOL             bAntiFraudModule;

} WFSCRDCAPS, *LPWFSCRDCAPS;

typedef struct _wfs_crd_cardunit
{
    USHORT           usNumber;
    LPSTR            lpszCardName;
    USHORT           usType;
    ULONG            ulInitialCount;
    ULONG            ulCount;
    ULONG            ulRetainCount;
    ULONG            ulThreshold;
    USHORT           usStatus;
    BOOL             bHardwareSensor;
} WFSCRDCARDUNIT, *LPWFSCRDCARDUNIT;

typedef struct _wfs_crd_cu_info
{
    USHORT           usCount;
    LPWFSCRDCARDUNIT *lppList;
} WFSCRDCUINFO, *LPWFSCRDCUINFO;

/*=================================================================*/
/* CRD Execute Command Structures */
/*=================================================================*/

typedef struct _wfs_crd_dispense
{
    USHORT           usNumber;
    BOOL             bPresent;
} WFSCRDDISPENSE, *LPWFSCRDDISPENSE;

typedef struct _wfs_crd_retain_card
{
    USHORT           usNumber;
} WFSCRDRETAINCARD, *LPWFSCRDRETAINCARD;

typedef struct _wfs_crd_reset
{
    USHORT           usAction;
} WFSCRDRESET, *LPWFSCRDRESET;

typedef struct _wfs_crd_set_guidlight
{
    WORD             wGuidLight;
    DWORD            dwCommand;
} WFSCRDSETGUIDLIGHT, *LPWFSCRDSETGUIDLIGHT;

typedef struct _wfs_crd_power_save_control
{
    USHORT           usMaxPowerSaveRecoveryTime;
} WFSCRDPOWERSAVECONTROL, *LPWFSCRDPOWERSAVECONTROL;
```

```
/*=================================================================*/
/* CRD Message Structures */
/*=================================================================*/

typedef struct _wfs_crd_media_detected
{
    WORD             wPosition;
    USHORT           usNumber;
} WFSCRDMEDIADETECTED, *LPWFSCRDMEDIADETECTED;

typedef struct _wfs_crd_cu_error
{
    WORD             wFailure;
    LPWFSCRDCARDUNIT lpCardUnit;
} WFSCRDCUERROR, *LPWFSCRDCUERROR;

typedef struct _wfs_crd_device_position
{
    WORD             wPosition;
} WFSCRDDEVICEPOSITION, *LPWFSCRDDEVICEPOSITION;

typedef struct _wfs_crd_power_save_change
{
    USHORT           usPowerSaveRecoveryTime;
} WFSCRDPOWERSAVECHANGE, *LPWFSCRDPOWERSAVECHANGE;

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

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

#endif  /* __INC_XFSCRD__H */
```