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

WORKSHOP AGREEMENT

CWA 14050-5

November 2000

ICS 35.200; 35.240.40

Extensions for Financial Services (XFS) interface specification - Release 3.0 - Part 5: Cash Dispenser Device Class Interface

Ref. No CWA 14050-5:2000 E

Rue de Stassart, 36 • B-1050 Bruxelles
Tel : +32 2 550 08 11 • Fax : +32 2 550 08 19

Table of Contents

Foreword

This CWA is revision 3.0 of the XFS interface specification.

The move from an XFS 2.0 specification (CWA 13449) to a 3.0 specification has been prompted by a series of factors.

Initially, there has been a technical imperative to extend the scope of the existing specification of the XFS Manager to include new devices, such as the Card Embossing Unit.

Similarly, there has also been pressure, through implementation experience and the advance of the Microsoft technology, to extend the functionality and capabilities of the existing devices covered by the specification.

Finally, it is also clear that our customers and the market are asking for an update to a specification, which is now over 2 years old. Increasing market acceptance and the need to meet this demand is driving the Workshop towards this release.

The clear direction of the CEN/ISSS XFS Workshop, therefore, is the delivery of a new Release 3.0 specification based on a C API. It will be delivered with the promise of the protection of technical investment for existing applications and the design to safeguard future developments.

The CEN/ISSS XFS Workshop gathers suppliers as well as banks and other financial service companies. A list of companies participating in this Workshop and in support of this CWA is available from the CEN/ISSS Secretariat.

This CWA was formally approved by the XFS Workshop meeting on 2000-10-18. 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 3.0.

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 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: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 17: Printer Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 18: Identification Card Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 19: Cash Dispenser Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 20: PIN Keypad Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 21: Depository Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 22: Text Terminal Unit Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 23: Sensors and Indicators Unit Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 24: Camera Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 25: Identification Card Device Class Interface - PC/SC Integration Guidelines

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.cenorm.be/isss/Workshop/XFS.

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.

Revision History:

3.00	October 18, 2000	Addition of the reset command.
		Cash Units Manipulated enhancement.
		Count Command Addition, Coin Dispense Clarification
		Addition of rejected/retracted notes counter
		Addition of retract area structure
		Clarify handling of coins and bills, introduction of the term item instead of bill, clarification for mix tables, clarification for TEST_CASH_UNITS
		Other CDM error codes .
		For a detailed description see CWA 14050-19 CDM migration from version 2.00 to version 3.00, revision 1.00, October 18th 2000

1. Introduction

1.1 Background to Release 3.0

The CEN XFS Workshop is a continuation of the Banking Solution Vendors Council workshop and maintains a technical commitment to the Win 32 API. However, the XFS Workshop has extended the franchise of multi vendor software by encouraging the participation of both banks and vendors to take part in the deliberations of the creation of an industry standard. This move towards opening the participation beyond the BSVC's original membership has been very succesful with a current membership level of more than 20 companies.

The fundamental aims of the XFS Workshop are to promote a clear and unambiguous specification for both service providers and application developers. This has been achieved to date by sub groups working electronically and quarterly meetings.

The move from an XFS 2.0 specification to a 3.0 specification has been prompted by a series of factors. Initially, there has been a technical imperative to extend the scope of the existing specification of the XFS Manager to include new devices, such as the Card Embossing Unit.

Similarly, there has also been pressure, through implementation experience and the advance of the Microsoft technology, to extend the functionality and capabilities of the existing devices covered by the specification.

Finally, it is also clear that our customers and the market are asking for an update to a specification, which is now over 2 years old. Increasing market acceptance and the need to meet this demand is driving the Workshop towards this release.

The clear direction of the XFS Workshop, therefore, is the delivery of a new Release 3.0 specification based on a C API. It will be delivered with the promise of the protection of technical investment for existing applications and the design to safeguard future developments.

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 the Extensions for Financial Services 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 behaviour 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. Cash Dispensers

This specification describes the functionality of a XFS compliant Cash Dispenser Module (CDM) service provider. It defines the service-specific commands that can be issued to the service provider using the **WFSGetInfo, WFSAsyncGetInfo**, **WFSExecute** and **WFSAsyncExecute** functions.

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

This specification covers the dispensing of items. An "item" is defined as any media that can be dispensed and includes coupons, documents, bills and coins. However, if coins and bills are both to be dispensed separate service providers must be implemented for each.

All currency parameters in this specification are expressed as a quantity of minimum dispense units, as defined in the description of the WFS_INF_CDM_CURRENCY_EXP command (see Section 4.5).

There are two types of CDM: Self-Service CDM and Teller CDM. A Self-Service CDM operates in an automated environment, while a Teller CDM has an operator present. The functionality provided by the following commands is only applicable to a Teller CDM:

WFS_CMD_CDM_SET_TELLER_INFO
WFS_INF_CDM_ TELLER_INFO

It is possible for the CDM to be part of a compound device with the Cash In Module (CIM). This CIM\CDM combination is referred to throughout this specification as a "Cash Recycler". For details of the CIM interface see Ref. 3.

If the device is a Cash Recycler then, if cash unit exchanges are required on both interfaces, the exchanges cannot be performed concurrently. An exchange on one interface must be complete (the WFS_CMD_CDM_END_EXCHANGE must have completed) before an exchange can start on the other interface. The WFS_ERR_CDM_EXCHANGEACTIVE error code will be returned if the correct sequence is not adhered to. If the device has recycle units of multiple currencies and/or denominations, then the CDM interface should be used for exchange operations involving these cash units.

The Cash-Out cash unit counts will be available through the CDM interface and the Cash-In cash unit counts will be available through the CIM interface. Counts for recycle cash units are available through both interfaces. The event WFS_SRVE_CDM_COUNTS_CHANGED will be posted if an operation on the CIM interface effects the recycle cash unit counts which are available through the CDM interface.

The following commands on the CIM interface may affect the CDM counts:

WFS_CMD_CIM_CASH_IN
WFS_CMD_CIM_CASH_IN_ROLLBACK
WFS_CMD_CIM_RETRACT
WFS_CMD_CIM_SET_CASH_IN_UNIT_INFO
WFS_CMD_CIM_END_EXCHANGE
WFS_CMD_CIM_RESET
WFS_CMD_CIM_TEST_CASH_UNITS

3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer’s Reference, Revision 3.00, October 18, 2000
2. ISO 4217 at http://www.iso.ch/
3. XFS Cash In Module Device Class Interface, Programmer’s Reference, Revision 3.00, October 18, 2000

4. Info Commands

4.1 WFS_INF_CDM_STATUS

Description This command is used to obtain the status of the CDM. It may also return vendor-specific status information.

Input Param None.

Output Param `LPWFSCDMSTATUS    lpStatus;`

```
typedef struct _wfs_cdm_status
   {
   WORD              fwDevice;
   WORD              fwSafeDoor;
   WORD              fwDispenser;
   WORD              fwIntermediateStacker;
   LPWFSCDMOUTPOS *  lppPositions;
   LPSTR             lpszExtra;
   } WFSCDMSTATUS, *  LPWFSCDMSTATUS;
```

fwDevice
Supplies the state of the CDM. However, a *fwDevice* status of WFS_CDM_DEVONLINE does not necessarily imply that dispensing can take place: the value of the *fwDispenser* field must be taken into account and - for some vendors - the state of the safe door (*fwSafeDoor*) may also be relevant. The state of the CDM will have one of the following values:

Value	Meaning
WFS_CDM_DEVONLINE	The device is online. This is returned when the dispenser is present and operational.
WFS_CDM_DEVOFFLINE	The device is offline (e.g. the operator has taken the device offline by turning a switch or pulling out the device).
WFS_CDM_DEVPOWEROFF	The device is powered off or physically not connected.
WFS_CDM_DEVNODEVICE	The device is not 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_CDM_DEVHWERROR	The device is inoperable due to a hardware error.
WFS_CDM_DEVUSERERROR	The device is present but a person is preventing proper device operation.
WFS_CDM_DEVBUSY	The device is busy and unable to process an execute command at this time.

fwSafeDoor
Supplies the state of the safe door as one of the following values:

Value	Meaning
WFS_CDM_DOORNOTSUPPORTED	Physical device has no safe door or door state reporting is not supported.
WFS_CDM_DOOROPEN	Safe door is open.
WFS_CDM_DOORCLOSED	Safe door is closed.
WFS_CDM_DOORUNKNOWN	Due to a hardware error or other condition, the state of the door cannot be determined.

fwDispenser

Supplies the state of the dispenser's logical cash units as one of the following values:

Value	Meaning
WFS_CDM_DISPOK	All cash units present are in a good state.
WFS_CDM_DISPCUSTATE	The dispenser is operational, but one or more of the cash units is in a low, empty or inoperative condition. Items can still be dispensed from at least one of the cash units.
WFS_CDM_DISPCUSTOP	Due to a cash unit failure dispensing is impossible. The dispenser is operational, but no items can be dispensed because all of the cash units are in an empty or inoperative condition. This state also occurs when a reject/retract cash unit is full or no reject/retract cash unit is present, or an application lock is set on every cash unit.
WFS_CDM_DISPCUUNKNOWN	Due to a hardware error or other condition, the state of the cash units cannot be determined.

fwIntermediateStacker

Supplies the state of the intermediate stacker. These bills are typically present on the intermediate stacker as a result of a retract operation or because a dispense has been performed without a subsequent present. Possible values for this field are:

Value	Meaning
WFS_CDM_ISEMPTY	The intermediate stacker is empty.
WFS_CDM_ISNOTEMPTY	The intermediate stacker is not empty. The items have not been in customer access.
WFS_CDM_ISNOTEMPTYCUST	The intermediate stacker is not empty. The items have been in customer access. If the device is a recycler then the items on the intermediate stacker may be there as a result of a previous Cash-In operation.
WFS_CDM_ISNOTEMPTYUNK	The intermediate stacker is not empty. It is not known if the items have been in customer access
WFS_CDM_ISUNKNOWN	Due to a hardware error or other condition, the state of the intermediate stacker cannot be determined.
WFS_CDM_ISNOTSUPPORTED	The physical device has no intermediate stacker.

lppPositions

Pointer to a NULL terminated array of pointers to WFSCDMOUTPOS structures. There is one structure for each position to which items can be dispensed or presented:

```
typedef struct _wfs_cdm_position
{
WORD         fwPosition;
WORD         fwShutter;
WORD         fwPositionStatus;
WORD         fwTransport;
WORD         fwTransportStatus;
} WFSCDMOUTPOS, * LPWFSCDMOUTPOS;
```

fwPosition

Supplies the output position as one of the following values:

Value	Meaning
WFS_CDM_POSLEFT	Left output position.
WFS_CDM_POSRIGHT	Right output position.
WFS_CDM_POSCENTER	Center output position.
WFS_CDM_POSTOP	Top output position.
WFS_CDM_POSBOTTOM	Bottom output position.
WFS_CDM_POSFRONT	Front output position.
WFS_CDM_POSREAR	Rear output position.

fwShutter
Supplies the state of the shutter as one of the following values:

Value	Meaning
WFS_CDM_SHTCLOSED	The shutter is closed.
WFS_CDM_SHTOPEN	The shutter is opened.
WFS_CDM_SHTJAMMED	The shutter is jammed.
WFS_CDM_SHTUNKNOWN	Due to a hardware error or other condition, the state of the shutter cannot be determined.
WFS_CDM_SHTNOTSUPPORTED	The physical device has no shutter or shutter state reporting is not supported.

fwPositionStatus
Returns information regarding items which may be at the output position. If the device is a recycler it is possible that the output position will not be empty due to a previous Cash-In operation. The possible values of this field are:

Value	Meaning
WFS_CDM_PSEMPTY	The output position is empty.
WFS_CDM_PSNOTEMPTY	The output position is not empty.
WFS_CDM_PSUNKNOWN	Due to a hardware error or other condition, the state of the output position cannot be determined.
WFS_CDM_PSNOTSUPPORTED	The device is not capable of reporting whether or not items are at the output position.

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

Value	Meaning
WFS_CDM_TPOK	The transport is in a good state.
WFS_CDM_TPINOP	The transport is inoperative due to a hardware failure or media jam.
WFS_CDM_TPUNKNOWN	Due to a hardware error or other condition the state of the transport cannot be determined.
WFS_CDM_TPNOTSUPPORTED	The physical device has no transport or transport state reporting is not supported.

fwTransportStatus
Returns information regarding items which may be on the transport. If the device is a recycler device it is possible that the transport will not be empty due to a previous Cash-In operation. The possible values of this field are:

Value	Meaning
WFS_CDM_TPSTATEMPTY	The transport is empty.
WFS_CDM_TPSTATNOTEMPTY	The transport is not empty.
WFS_CDM_TPSTATNOTEMPTYCUST	Items which a customer has had access to are on the transport.
WFS_CDM_TPSTATNOTEMPTY_UNK	Due to a hardware error or other condition it is not known whether there are items on the transport.
WFS_CDM_TPSTATNOTSUPPORTED	The device is not capable of reporting whether items are on the transport.

lpszExtra
A string of vendor-specific information consisting of “*key=value*” sub-strings. Each sub-string is null-terminated, with the final sub-string terminating with two null characters.

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

Comments Applications which rely on the *lpszExtra* parameter may not be device or vendor-independent.

4.2 WFS_INF_CDM_CAPABILITIES

Description This command retrieves the capabilities of the CDM. It may also return vendor specific capability information. The intermediate stacker and the transport are treated as separate areas. Some devices may have the capability to move items from the cash units to the intermediate stacker while there are items on the transport. Similarly some devices may be able to retract items to the transport or the cash units while there are items on the intermediate stacker.

Input Param None.

Output Param `LPWFSCDMCAPS lpCaps;`

```
typedef struct _wfs_cdm_caps
   {
   WORD         wClass;
   WORD         fwType;
   WORD         wMaxDispenseItems;
   BOOL         bCompound;
   BOOL         bShutter;
   BOOL         bShutterControl;
   WORD         fwRetractAreas;
   WORD         fwRetractTransportActions;
   WORD         fwRetractStackerActions;
   BOOL         bSafeDoor;
   BOOL         bCashBox;
   BOOL         bIntermediateStacker;
   BOOL         bItemsTakenSensor;
   WORD         fwPositions;
   WORD         fwMoveItems;
   WORD         fwExchangeType;
   LPSTR        lpszExtra;
   } WFSCDMCAPS, * LPWFSCDMCAPS;
```

wClass
Specifies the service class. Value is:
WFS_SERVICE_CLASS_CDM

fwType
Supplies the type of CDM as one of the following values:

Value	Meaning
WFS_CDM_TELLERBILL	The CDM is a Teller Bill Dispenser.
WFS_CDM_SELFSERVICEBILL	The CDM is a Self Service Bill Dispenser.
WFS_CDM_TELLERCOIN	The CDM is a Teller Coin Dispenser.
WFS_CDM_SELFSERVICECOIN	The CDM is a Self Service Coin Dispenser.

wMaxDispenseItems
Supplies the maximum number of items that can be dispensed in a single dispense operation. If no limit applies this value will be 0 – in this case, if an attempt is made to dispense more items than the hardware limitations will allow, the service provider will implement the dispense as a series of sub-dispense operations [see section .Sub-Dispensing Command Flow].

bCompound
Specifies whether the CDM is part of a compound device. If the CDM is part of a compound device with a CIM then this combination can be referred to as a recycler. In this case, no information on Cash-In cash units will be supplied via the CDM interface. The CDM interface will however supply information on shared retract or reject cash units and recycler cash units.

bShutter
Specifies whether or not the commands WFS_CMD_CDM_OPEN_SHUTTER and WFS_CMD_CDM_CLOSE_SHUTTER are supported.

bShutterControl
If set to TRUE the shutter is controlled implicitly by the service provider. If set to FALSE the shutter must be controlled explicitly by the application using the WFS_CMD_CDM_OPEN_SHUTTER and the WFS_CMD_CDM_CLOSE_SHUTTER commands. This field is always set to TRUE if the device has no shutter. This field applies to all shutters and all output positions.

fwRetractAreas
Specifies the area to which items may be retracted as a combination of the following flags:

Value	Meaning
WFS_CDM_RA_RETRACT	The items may be retracted to the retract cash unit.
WFS_CDM_RA_TRANSPORT	The items may be retracted to the transport.
WFS_CDM_RA_STACKER	The items may be retracted to the intermediate stacker.
WFS_CDM_RA_REJECT	The items may be retracted to the reject cash unit.
WFS_CDM_RA_NOTSUPP	The CDM does not have the ability to retract.

fwRetractTransportActions
Specifies the actions which may be performed on items which have been retracted to the transport. This field will be a combination of the following flags:

Value	Meaning
WFS_CDM_PRESENT	The items may be presented.
WFS_CDM_RETRACT	The items may be retracted to a retract cash unit.
WFS_CDM_REJECT	The items may be rejected to a reject bin.
WFS_CDM_NOTSUPP	The CDM does not have the ability to retract from the transport.

fwRetractStackerActions
Specifies the actions which may be performed on items which have been retracted to the stacker. If the device does not have a retract capability this value will be WFS_CDM_NOTSUPP. Otherwise it will be a combination of the following flags:

Value	Meaning
WFS_CDM_PRESENT	The items may be presented.
WFS_CDM_RETRACT	The items may be retracted to a retract cash unit.
WFS_CDM_REJECT	The items may be rejected to a reject bin.
WFS_CDM_NOTSUPP	The CDM does not have the ability to retract from the stacker.

bSafedoor
Specifies whether or not the WFS_CMD_CDM_OPEN_SAFE_DOOR command is supported.

bCashBox
This field is only applicable to CDM types WFS_CDM_TELLERBILL and WFS_CDM_TELLERCOIN. It specifies whether or not Tellers have been assigned a Cash Box.

bIntermediateStacker
Specifies whether or not the CDM supports stacking items to an intermediate position before the items are moved to the exit position. If this value is TRUE, the parameter *bPresent* of the WFS_CMD_CDM_DISPENSE command can be set to FALSE [see Section WFS_CMD_CDM_DISPENSE].

bItemsTakenSensor
Specifies whether the CDM can detect when items at the exit position are taken by the user. If set to TRUE the service provider generates an accompanying WFS_SRVE_CDM_ITEMS_TAKEN event. If set to FALSE this event is not generated. This field applies to all output positions.

fwPositions
Specifies the CDM output positions which are available as a combination of the following flags:

Value	Meaning
WFS_CDM_POSLEFT	The CDM has a left output position.
WFS_CDM_POSRIGHT	The CDM has a right output position.
WFS_CDM_POSCENTER	The CDM has a center output position.
WFS_CDM_POSTOP	The CDM has a top output position.
WFS_CDM_POSBOTTOM	The CDM has a bottom output position.
WFS_CDM_POSFRONT	The CDM has a front output position.
WFS_CDM_POSREAR	The CDM has a rear output position.

fwMoveItems
Specifies the CDM move item options which are available as a combination of the following flags:

Value	Meaning
WFS_CDM_FROMCU	The CDM can move items from the cash units to the intermediate stacker while there are items on the transport.
WFS_CDM_TOCU	The CDM can retract items to the cash units while there are items on the intermediate stacker.
WFS_CDM_TOTRANSPORT	The CDM can retract items to the transport while there are items on the intermediate stacker.

fwExchangeType
Specifies the type of cash unit exchange operations supported by the CDM as a combination of the following flags:

Value	Meaning
WFS_CDM_EXBYHAND	The CDM supports manual replenishment either by filling the cash unit by hand or by replacing the cash unit.
WFS_CDM_EXTOCASSETTES	The CDM supports moving items from the replenishment cash unit to another cash unit.

lpszExtra
A string of vendor-specific information consisting of "*key=value*" sub-strings. Each sub-string is null-terminated, with the final sub-string terminating with two null characters.

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

Comments Applications which rely on the *lpszExtra* parameter may not be device or vendor-independent.

4.3 WFS_INF_CDM_CASH_UNIT_INFO

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

Where a logical cash unit is configured but there is no corresponding physical cash unit currently present in the device, information about the missing cash unit will still be returned in the *lppList* field of the output parameter. The status of the cash unit will be reported as WFS_CDM_STATCUMISSING.

It is possible that one logical cash unit may be associated with more than one physical cash unit. In this case, the number of cash unit structures returned in *lpCashUnitInfo* will reflect the number of logical cash units in the CDM. That is, if a system contains four physical cash units but two of these are treated as one logical cash unit, *lpCashUnitInfo* will contain information about the three logical cash units and a *usCount* of 3. Information about the physical cash unit(s) associated with a logical cash unit is contained in the WFSCDMCASHUNIT structure representing the logical cash unit.

It is also possible that multiple logical cash units may be associated with one physical cash unit. This should only occur if the physical cash unit is capable of handling this situation, i.e. if it can store multiple denominations and report meaningful count and replenishment information for each denomination. In this case the information returned in *lpCashUnitInfo* will again reflect the number of logical cash units in the CDM.

Logical Types
A cash unit may have a logical type. A logical type is based on the value of the following fields of the WFSCDMCASHUNIT structure:
lpCashUnitName
usType
cCurrencyID

ulValues
A logical type of cash unit may be associated with more than one physical cash unit.
The logical type is distinct from the logical number (*usNumber),* i.e. *usNumber* does not refer to the logical cassette type.

Counts
The values of the following fields of the WFSCDMCASHUNIT and WFSCDMPHCU structures:
ulCount
ulRejectCount
are software counts and therefore may not represent the actual number of items in the cash unit.

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

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

The application can check if the device has this capability by querying the *bHardwareSensors* field of the physical cash unit structure. If any of the physical cash units associated with the logical cash unit have this capability, then threshold events based on hardware sensors can be triggered.

In the situation where the cash unit is associated with multiple physical cash units, if the service provider has the capability, the WFS_SRVE_CDM_CASHUNITINFOCHANGED event may be generated when any of the physical cash units reaches the threshold. When the final physical cash unit reaches the threshold, the WFS_USRE_CDM_CASHUNITTHRESHOLD event will be generated.

Exchanges
If a physical cash unit is removed when the device is not in the exchange state the status of the physical cash unit will be set to WFS_CDM_STATMANIP and the values of the physical cash unit prior to its' removal will be returned in any subsequent WFS_INF_CDM_CASH_UNIT_INFO command. The physical cash unit will not be used in any operation. The application must perform an exchange operation specifying the new values for the physical cash unit in order to recover the situation.

Recyclers
Through the CDM interface a service provider does not report cash-in cash units and through the CIM interface it does not report cash out cash units. But both device classes report the recycling cash units (WFS_CDM_TYPERECYCLING).

Input Param None.

Output Param `LPWFSCDMCUINFO    lpCashUnitInfo;`

```
typedef struct _wfs_cdm_cu_info
  {
  USHORT              usTellerID;
  USHORT              usCount;
  LPWFSCDMCASHUNIT *  lppList;
  } WFSCDMCUINFO, * LPWFSCDMCUINFO;
```

usTellerID
This field is not used in this command and is always 0.

usCount
Specifies the number of cash unit structures returned.

lppList
Pointer to an array of pointers to cash unit structures:

```
typedef struct _wfs_cdm_cashunit
   {
   USHORT          usNumber;
   USHORT          usType;
   LPSTR           lpszCashUnitName;
   CHAR            cUnitID[5];
   CHAR            cCurrencyID[3];
   ULONG           ulValues;
   ULONG           ulInitialCount;
   ULONG           ulCount;
   ULONG           ulRejectCount;
   ULONG           ulMinimum;
   ULONG           ulMaximum;
   BOOL            bAppLock;
   USHORT          usStatus;
   USHORT          usNumPhysicalCUs;
   LPWFSCDMPHCU  * lppPhysical;
   } WFSCDMCASHUNIT, * LPWFSCDMCASHUNIT;
```

usNumber
Index number of the cash 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.

usType
Type of cash unit. Possible values are:

Value	Meaning
WFS_CDM_TYPENA	Not applicable. Typically means cash unit is missing.
WFS_CDM_TYPEREJECTCASSETTE	Reject cash unit.
WFS_CDM_TYPEBILLCASSETTE	Cash unit containing bills.
WFS_CDM_TYPECOINCYLINDER	Coin cylinder.
WFS_CDM_TYPECOINDISPENSER	Coin dispenser as a whole unit.
WFS_CDM_TYPERETRACTCASSETTE	Retract cash unit.
WFS_CDM_TYPECOUPON	Cash unit containing coupons or advertising material.
WFS_CDM_TYPEDOCUMENT	Cash unit containing documents.
WFS_CDM_TYPEREPCONTAINER	Replenishment container. A cash unit can be refilled from a replenishment container.
WFS_CDM_TYPERECYCLING	Recycling cash unit. This unit is only present when the device is a compound device with a CIM.

cUnitID
The Cash Unit Identifier.

lpszCashUnitName
A name which helps to identify the logical type of the cash unit. This is especially useful in the case of cash units of type WFS_CDM_TYPEDOCUMENT where different documents can have the same currency and value. For example, travellers cheques and bank cheques may have the same currency and value but still need to be identifiable as different types of document. Where this value is not relevant (e.g. in bill cash units) the pointer will be NULL.

cCurrencyID
A three character array storing the ISO format [Ref. 2] Currency ID. This value will be an array of three ASCII 0x20h characters for cash units which contain items of more than one currency type or items to which currency is not applicable. If the *usStatus* field for this cash unit is WFS_CDM_STATCUNOVAL it is the responsibility of the application to assign a value to this field.

ulValues
Supplies the value of a single item in the cash unit. This value is expressed in minimum dispense units [see Section WFS_INF_CDM_CURRENCY_EXP]. If the *cCurrencyID* field for this cash unit is empty, then this field will contain 0. If the *usStatus* field for this cash unit is WFS_CDM_STATCUNOVAL it is the responsibility of the application to assign a value to this field.

ulInitialCount
Initial number of items contained in the cash unit. This value is persistent. If the cash unit is a recycle cash unit then this value will be incremented as a result of a Cash-In operation.

ulCount
The number of items inside all the physical cash units associated with this cash unit, plus any items from these physical cash units not yet presented to the customer. This count is decremented when the items are either presented to the customer or rejected.

If the cash unit is a recycle cash unit then this value will be incremented as a result of a Cash-In operation.

Note that for a reject cash units, this value is unreliable, since the typical reason for dumping items to the reject cash unit is a suspected count failure. For a retract cash unit this value specifies the number of retracts.

If this value reaches 0 it will not decrement further but will remain at 0. This value is persistent.

ulRejectCount
The number of items from this cash unit which are in the reject bin. This value may be unreliable, since the typical reason for dumping items to the reject cash unit is a suspected pick failure. This value is persistent.

ulMinimum
This field is not applicable to Retract and Reject Cash Units. For all other cash units, when *ulCount* reaches this value the threshold event WFS_USRE_CDM_CASHUNITTHRESHOLD will be generated. If this value is non-0 then hardware sensors in the device do not trigger threshold events.

ulMaximum
This field is only applicable to Retract and Reject Cash Units. When *ulCount* reaches this value the threshold event WFS_USRE_CDM_CASHUNITTHRESHOLD will be generated. If this value is non-0 then hardware sensors in the device do not trigger threshold events.

bAppLock
This field does not apply to reject or retract cash units. If this value is TRUE items cannot be dispensed from the cash unit. If this value is TRUE and the application attempts to dispense from the cash unit a WFS_EXEE_CDM_CASHUNITERROR event will be generated and a WFS_ERR_CDM_CASHUNITERROR code will be returned.

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

Value	Meaning
WFS_CDM_STATCUOK	The cash unit is in a good state.
WFS_CDM_STATCUFULL	The cash unit is full.
WFS_CDM_STATCUHIGH	The cash unit is almost full (i.e. nearing the threshold defined by *ulMaximum*).
WFS_CDM_STATCULOW	The cash unit is almost empty (i.e. nearing the threshold defined by *ulMinimum*).
WFS_CDM_STATCUEMPTY	The cash unit is empty.
WFS_CDM_STATCUINOP	The cash unit is inoperative.
WFS_CDM_STATCUMISSING	The cash unit is missing.
WFS_CDM_STATCUNOVAL	The values of the specified cash unit are not available.
WFS_CDM_STATCUNOREF	There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.
WFS_CDM_STATCUMANIP	The cash unit has been changed when the device was not in the exchange state. This cash unit cannot be dispensed from.

usNumPhysicalCUs
The number of physical cash unit structures returned in the following *lppPhysical* array. This number must be at least 1.

lppPhysical
Pointer to an array of pointers to physical cash unit structures:

```
typedef struct _wfs_cdm_physicalcu
    {
    LPSTR     lpPhysicalPositionName;
    CHAR      cUnitID[5];
    ULONG     ulInitialCount;
    ULONG     ulCount;
    ULONG     ulRejectCount;
    ULONG     ulMaximum;
    USHORT    usPStatus;
    BOOL      bHardwareSensor;
    } WFSCDMPHCU, * LPWFSCDMPHCU;
```

lpPhysicalPositionName
A name identifying the physical location of the cash unit within the CDM. This field can be used by CDMs which are compound with a CIM to identify shared cash units.

cUnitID
A 5 character array uniquely identifying the physical cash unit.

ulInitialCount
Initial number of items contained in the cash unit. If the cash unit is a recycle cash unit then this count may be incremented as a result of a Cash-In operation. This value is persistent.

ulCount
Actual count of items in the physical cash unit. This count is decremented whenever a bill leaves the physical cash unit for any reason. This count may be incremented if the cash unit is a recycle cash unit. This value is persistent.

ulRejectCount
The number of items from this cash unit which are in the reject bin. This value may be unreliable, since the typical reason for dumping items to the reject cash unit is a suspected pick failure. This value is persistent.

ulMaximum
The maximum number of items the cash unit can hold. This is only for informational purposes. No threshold event WFS_USRE_CDM_CASHUNITTHRESHOLD will be generated.

usPStatus
Supplies the status of the physical cash unit as one of the following values:

Value	Meaning
WFS_CDM_STATCUOK	The cash unit is in a good state.
WFS_CDM_STATCUFULL	The cash unit is full.
WFS_CDM_STATCUHIGH	The cash unit is almost full (threshold defined by ulMaximum).
WFS_CDM_STATCULOW	The cash unit is almost empty (threshold defined by ulMinimum).
WFS_CDM_STATCUEMPTY	The cash unit is empty.
WFS_CDM_STATCUINOP	The cash unit is inoperative.
WFS_CDM_STATCUMISSING	The cash unit is missing.
WFS_CDM_STATCUNOVAL	The values of the specified cash unit are not available.
WFS_CDM_STATCUNOREF	There is no reference value available for the notes in this cash unit. The cash unit has not been calibrated.
WFS_CDM_STATCUMANIP	The cash unit has been changed when the device was not in the exchange state. This cash unit cannot be dispensed from.

bHardwareSensor
Specifies whether or not threshold events can be generated based on hardware sensors in the device. If this value is TRUE for any of the physical cash units related to a logical cash unit then threshold events may be generated based on hardware sensors as opposed to logical counts.

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

Comments None.

4.4 WFS_INF_CDM_TELLER_INFO

Description This command only applies to Teller CDMs. It allows the application to obtain counts for each currency assigned to the teller. These counts represent the total amount of currency dispensed by the teller in all transactions.

This command also enables the application to obtain the position assigned to each Teller. If the input parameter is NULL, this command will return information for all Tellers and all currencies. The teller information is persistent.

Input Param

```
LPWFSCDMTELLERINFO lpTellerInfo;

typedef struct _wfs_cdm_teller_info
    {
    USHORT        usTellerID;
    CHAR          cCurrencyID[3];
    } WFSCDMTELLERINFO, *LPWFSCDMTELLERINFO;
```

usTellerID
Identification of the teller. If the value of *usTellerID* is not valid the error WFS_ERR_CDM_INVALIDTELLERID is reported.

cCurrencyID
Three character ISO format currency identifier [Ref 2]

This parameter can be an array of three ASCII 0x20h characters. In this case information on all currencies will be returned.

Output Param `LPWFSCDMTELLERDETAILS * lpTellerDetails;`

Pointer to a null-terminated array of pointers to teller info structures.

```
typedef struct _wfs_cdm_teller_details
   {
   USHORT                 usTellerID;
   ULONG                  ulInputPosition;
   WORD                   fwOutputPosition
   LPWFSCDMTELLERTOTALS*  lppTellerTotals;
   } WFSCDMTELLERDETAILS, * LPWFSCDMTELLERDETAILS;
```

usTellerID
Identification of the teller.

ulInputPosition
The input position assigned to the teller for cash entry. This is only for compatibility except when the device is a compound device. The value is specified by one of the following values:

Value	Meaning
WFS_CDM_POSNULL	No position is assigned to the Teller.
WFS_CDM_POSINLEFT	Left position is assigned to the Teller.
WFS_CDM_POSINRIGHT	Right position is assigned to the Teller.
WFS_CDM_POSINCENTER	Center position is assigned to the Teller.
WFS_CDM_POSINTOP	Top position is assigned to the Teller.
WFS_CDM_POSINBOTTOM	Bottom position is assigned to the Teller.
WFS_CDM_POSINFRONT	Front position is assigned to the Teller.
WFS_CDM_POSINREAR	Rear position is assigned to the Teller.

fwOutputPosition
The output position from which cash is presented to the teller. The value is specified by one of the following values:

Value	Meaning
WFS_CDM_POSNULL	No position is assigned to the Teller.
WFS_CDM_POSLEFT	Left position is assigned to the Teller.
WFS_CDM_POSRIGHT	Right position is assigned to the Teller.
WFS_CDM_POSCENTER	Center position is assigned to the Teller.
WFS_CDM_POSTOP	Top position is assigned to the Teller.
WFS_CDM_POSBOTTOM	Bottom position is assigned to the Teller.
WFS_CDM_POSFRONT	Front position is assigned to the Teller.
WFS_CDM_POSREAR	Rear position is assigned to the Teller.

lppTellerTotals
Pointer to a null-terminated array of pointers to teller total structures.

```
typedef struct _wfs_cdm_teller_totals
  {
  CHAR         cCurrencyID[3];
  ULONG        ulItemsReceived;
  ULONG        ulItemsDispensed;
  ULONG        ulCoinsReceived;
  ULONG        ulCoinsDispensed;
  ULONG        ulCashBoxReceived;
  ULONG        ulCashBoxDispensed;
  } WFSCDMTELLERTOTALS, * LPWFSCDMTELLERTOTALS
```

cCurrencyID
Three character ISO format currency identifier [Ref. 2].

ulItemsReceived
The total amount of items (other than coins) of the specified currency accepted. The amount is expressed in minimum dispense units (see WFS_INF_CDM_CURRENCY_EXP).

ulItemsDispensed
The total amount of items (other than coins) of the specified currency dispensed. The amount is expressed in minimum dispense units (see WFS_INF_CDM_CURRENCY_EXP).

ulCoinsReceived
The total amount of coin currency accepted. The amount is expressed in minimum dispense units (see WFS_INF_CDM_CURRENCY_EXP).

ulCoinsDispensed
The total amount of coin currency dispensed. The amount is expressed in minimum dispense units (see WFS_INF_CDM_CURRENCY_EXP).

ulCashBoxReceived
The total amount of cash box currency accepted. The amount is expressed in minimum dispense units (see WFS_INF_CDM_CURRENCY_EXP).

ulCashBoxDispensed
The total amount of cash box currency dispensed. The amount is expressed in minimum dispense units (see WFS_INF_CDM_CURRENCY_EXP).

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_CDM_INVALIDCURRENCY	Specified currency not currently available
WFS_ERR_CDM_INVALIDTELLERID	Invalid Teller ID

Comments None.

4.5 WFS_INF_CDM_CURRENCY_EXP

Description This command returns each exponent assigned to each currency known to the service provider.

Input Param None.

Output Param `LPWFSCDMCURRENCYEXP *   lppCurrencyExp;`

Pointer to a null-terminated array of pointers to currency exponent structures:

```
typedef struct _wfs_cdm_currency_exp
   {
   CHAR         cCurrencyID[3];
   SHORT        sExponent;
   } WFSCDMCURRENCYEXP, *LPWFSCDMCURRENCYEXP;
```

cCurrencyID
Currency identifier in ISO 4217 format [see Ref 2].

sExponent
Currency exponent in ISO 4217 format [see Ref. 2].

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

Comments For each currency ISO 4217 defines the currency identifier (a three character code) and a currency unit (e.g., German mark, Italian lira). In the interface defined by this specification, every money amount is specified in terms of multiples of the minimum dispense unit, which is equal to the currency unit times the currency exponent. Thus an amount parameter relates to the actual cash amount as follows:

<cash_amount> = <money_amount_parameter> * 10^<sExponent>

Example #1 — Germany

Currency identifier is ‘DEM’

Currency unit is 1 German mark (= 100 pfennig)

A service provider is developed for an ATM that can dispense coins down to one pfennig. The currency exponent (*sExponent*) is set to -2 (minus two), so the minimum dispense unit is one pfennig (1 * 10^-2 mark); all amounts at the XFS interface are in pfennigs. Thus a money amount parameter of 10050 is 100 marks and 50 pfennig.

Example #2 — Italy

Currency identifier is 'LIT'

Currency unit is 1 Italian lira

A service provider is required to dispense a minimum amount of 100 lire. The currency exponent (*sExponent*) is set to +2 (plus two), so the minimum dispense unit is 100 lire; all amounts at the XFS interface are in multiples of 100 lire. Thus a amount parameter of 150 is 15000 lire.

4.6 WFS_INF_CDM_MIX_TYPES

Description This command is used to obtain a list of supported mix algorithms and available house mix tables.

Input Param None.

Output Param `LPWFSCDMMIXTYPE * lppMixTypes;`

Pointer to a null-terminated array of pointers to mix type structures:

```
typedef struct _wfs_cdm_mix_type
   {
   USHORT       usMixNumber;
   USHORT       usMixType;
   USHORT       usSubType;
   LPSTR        lpszName;
   } WFSCDMMIXTYPE, *LPWFSCDMMIXTYPE;
```

usMixNumber
Number identifying the mix algorithm or the house mix table. This number can be passed to the WFS_INF_CDM_MIX_TABLE, WFS_CMD_CDM_DISPENSE and WFS_CMD_CDM_DENOMINATE commands.

usMixType
Specifies whether the mix type is an algorithm or a house mix table. Possible values are:

Value	Meaning
WFS_CDM_MIXALGORITHM	Mix algorithm.
WFS_CDM_MIXTABLE	Mix table.

usSubType
Contains a vendor-defined number that identifies the type of algorithm or table. Individual vendor-defined mix algorithms are defined above hexadecimal 7FFF. Mix algorithms which are provided by the service provider are in the range hexadecimal 8000 - 8999. Application defined mix algorithms start at hexadecimal 9000. All numbers below 8000 hexadecimal are reserved. Predefined values are:

Value	Meaning
WFS_CDM_MIX_MINIMUM_NUMBER_OF_BILLS	Select a mix requiring the minimum possible number of items.
WFS_CDM_MIX_EQUAL_EMPTYING_OF_CASH_UNITS	The denomination is selected based upon criteria which ensure that over the course of its operation the CDM cash units will empty as far as possible at the same rate and will therefore go LOW and then EMPTY at approximately the same time.
WFS_CDM_MIX_MAXIMUM_NUMBER_OF_CASH_UNITS	The denomination will be selected based upon criteria which ensures the maximum number of different value items are dispensed.

lpszName
Points to the name of the table/algorithm used.

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

Comments None.

4.7 WFS_INF_CDM_MIX_TABLE

Description This command is used to obtain the house mix table specified by the supplied mix number.

Input Param
```
LPUSHORT     lpusMixNumber;
```

lpusMixNumber
Points to the number of the requested house mix table.

Output Param
```
LPWFSCDMMIXTABLE  lpMixTable;

typedef struct _wfs_cdm_mix_table
    {
    USHORT           usMixNumber;
    LPSTR            lpszName;
    USHORT           usRows;
    USHORT           usCols;
    LPULONG          lpulMixHeader;
    LPWFSCDMMIXROW * lppMixRows;
    } WFSCDMMIXTABLE, *LPWFSCDMMIXTABLE;
```

usMixNumber
Number identifying the house mix table.

lpszName
Points to the name of the table.

usRows
Number of rows in the house mix table. There is at least one row for each distinct total amount to be denominated. If there is more than one row for an amount the first row is taken that is dispensable according to the current status of the cash units.

usCols
Number of columns in the house mix table. There is one column for each distinct item value included in the mix.

lpulMixHeader
Pointer to an array of length *usCols* of unsigned longs; each element defines the value of the item corresponding to its respective column. (See WFS_INF_CDM_CURRENCY_EXP)

lppMixRows
Pointer to an array (of length *usRows*) of pointers to WFSCDMMIXROW structures:

```
typedef struct _wfs_cdm_mix_row
    {
    ULONG              ulAmount;
    LPUSHORT           lpusMixture;
    } WFSCDMMIXROW, *LPWFSCDMMIXROW;
```

ulAmount
Amount denominated by this mix row (See WFS_INF_CDM_CURRENCY_EXP).

lpusMixture
Pointer to a mix row, an array of length *usCols* of unsigned integers; each element defines the quantity of each item denomination in the mix used in the denomination of *ulAmount*

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_CDM_INVALIDMIXNUMBER	The *lpusMixNumber* parameter does not correspond to a defined mix table.

4.8 WFS_INF_CDM_PRESENT_STATUS

Description This command is used to obtain the status of the most recent attempt to present items to the customer. The items may have been presented as a result of the WFS_CMD_CDM_PRESENT or WFS_CMD_CDM_DISPENSE command.

This value is persistent and is valid until the next time an attempt is made to present items to the customer.

Input Param LPWORD lpfwPosition;

lpfwPosition
Specifies the output position the items were presented or dispensed to as one of the following values:

Value	Meaning
WFS_CDM_POSNULL	The items were presented according to the default configuration.
WFS_CDM_POSLEFT	The items were presented to the left output position.
WFS_CDM_POSRIGHT	The items were presented to the right output position.
WFS_CDM_POSCENTER	The items were presented to the center output position.
WFS_CDM_POSTOP	The items were presented to the top output position.
WFS_CDM_POSBOTTOM	The items were presented to the bottom output position.
WFS_CDM_POSFRONT	The items were presented to the front output position.
WFS_CDM_POSREAR	The items were presented to the rear output position.

Output Param `LPWFSCDMPRESENTSTATUS  lpPresentStatus;`

```
typedef struct _wfs_cdm_present_status
  {
  LPWFSCDMDENOMINATION   lpDenomination;
  WORD                   wPresentState;
  LPSTR                  lpszExtra;
  } WFSCDMPRESENTSTATUS, *LPWFSCDMPRESENTSTATUS;
```

lpDenomination
Pointer to a WFSCDMDENOMINATION structure which contains the number of items from each cash unit. For a description of the WFSCDMDENOMINATION structure see the definition of the command WFS_CMD_CDM_DENOMINATE.

wPresentState
Supplies the status of the last dispense or present operation. Possible values are:

Value	Meaning
WFS_CDM_PRESENTED	The items were presented. This status is set as soon as the customer has access to the items.
WFS_CDM_NOTPRESENTED	The customer did not have access to the items.
WFS_CDM_UNKNOWN	It is not known if the customer had access to the items.

lpszExtra
A string of vendor-specific information consisting of "*key=value*" sub-strings. Each sub-string is null-terminated, with the final sub-string terminating with two null characters.

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_CDM_DENOMINATE

Description This command provides a denomination. A denomination specifies the number of items which are required from each cash unit in order to satisfy a given amount. The denomination depends upon the currency, the mix algorithm and any partial denomination supplied by the application.

This command can also be used to validate that any denomination supplied by the application can be dispensed.

If items of differing currencies are to be included in the same denomination then the currency field must be an array of three ASCII 0x20h characters, the amount must be 0 and the mix number must be WFS_CDM_INDIVIDUAL. However, these restrictions do not apply if a single currency is combined with non-currency items, such as coupons.

If the *bCashBox* field of the WFSCDMCAPS structure returned by the WFS_INF_CDM_CAPABILITIES command is TRUE then, if the entire denomination cannot be satisfied, a partial denomination will be returned with the remaining amount to be supplied from the Teller's cash box.

This command can be used in four different ways:

1. In order to check that it is possible to dispense a given denomination. The input parameters to the command are currency and denomination, with a mix number of WFS_CDM_INDIVIDUAL and an amount of 0. If items of differing currencies are to be dispensed then the currency field should be an array of three ASCII 0x20h characters.
2. In order to validate that a given amount matches a given denomination and that it is possible to dispense the denomination. The input parameters to the command should be amount and denomination, with a mix number of WFS_CDM_INDIVIDUAL
3. In order to obtain a denomination of a given amount. The input parameters supplied should be amount, currency and mix number.
4. In order to complete a partial denomination of a given amount. In this case the input parameters to the command should be currency, amount, mix number and either a partially specified denomination or a minimum amount from the cash box. A completed denomination is returned. *ulCashBox* of the denomination structure may be updated as a result of this command.

Input Param `LPWFSCDMDENOMINATE lpDenominate;`

```
typedef struct _wfs_cdm_denominate
   {
   USHORT                 usTellerID;
   USHORT                 usMixNumber;
   LPWFSCDMDENOMINATION   lpDenomination;
   } WFSCDMDENOMINATE, * LPWFSCDMDENOMINATE;
```

usTellerID
Identification of teller. This parameter is ignored if the device is a Self-Service CDM.

usMixNumber
Mix algorithm or house mix table to be used.

lpDenomination
Pointer to a WFSCDMDENOMINATION structure, describing the contents of the denomination operation.

```
typedef struct _wfs_cdm_denomination
   {
   CHAR          cCurrencyID[3];
   ULONG         ulAmount;
   USHORT        usCount;
   LPULONG       lpulValues;
   ULONG         ulCashBox;
   } WFSCDMDENOMINATION, * LPWFSCDMDENOMINATION;
```

cCurrencyID
Identification of currency in ISO format [see Ref. 2]. Where the denomination contains multiple currencies this field should be set to three ASCII 0x20 characters.

ulAmount
The amount to be denominated or dispensed. Where the denomination contains multiple currencies this value is 0.

usCount
The size of the *lpulValues* list. This *usCount* is the same as the *usCount* returned from the last WFS_INF_CDM_CASH_UNIT_INFO command or set by the last WFS_CMD_CDM_SET_CASH_UNIT_INFO or WFS_CMD_CDM_END_EXCHANGE commands. If this value is not required because a mix algorithm is used then the *usCount* can be set to 0.

If the application passes in an invalid *usCount* the service provider should return a WFS_ERR_INVALID_DATA return code.

lpulValues
Pointer to an array of ULONGs. This list specifies the number of items to take from each of the cash units. This list corresponds to the array of cash unit structures returned to the last WFS_INF_CDM_CASH_UNIT_INFO command or set by the last WFS_CMD_CDM_SET_CASH_UNIT_INFO or WFS_CMD_CDM_END_EXCHANGE commands. The first value in the array is related to the cash structure with the index number 1.

This array contains a field for each possible Cash Unit. If a Cash Unit is not required in the denomination it's corresponding field in this array should be set to zero.

If the application does not wish to specify a denomination, it should set the *lpulValues* pointer to NULL.

ulCashBox
Only applies to Teller CDM devices. Amount to be paid from the teller's cash box.

Output Param `LPWFSCDMDENOMINATION    lpDenomination;`

For a description see the input structure.

Where mixed currencies are being denominated the *ulAmount* field in the returned denomination structure will be 0 and the *cCurrency* field will be set to three ASCII 0x20 characters.

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_CDM_INVALIDCURRENCY	There are no cash units in the CDM of the currency specified in the *cCurrency* field of the input parameter.
WFS_ERR_CDM_INVALIDTELLERID	Invalid Teller ID.
WFS_ERR_CDM_CASHUNITERROR	There is a problem with a cash unit. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.
WFS_ERR_CDM_INVALIDDENOMINATION	The *usMixNumber* is WFS_CDM_INDIVIDUAL and the sum of the values for cashbox and denomination was greater than the amount specified.
WFS_ERR_CDM_INVALIDMIXNUMBER	Unknown mix algorithm.
WFS_ERR_CDM_NOCURRENCYMIX	The cash units specified in the denomination were not all of the same currency.
WFS_ERR_CDM_NOTDISPENSABLE	The amount is not dispensable by the CDM.
WFS_ERR_CDM_TOOMANYITEMS	The request requires too many items to be dispensed.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM is in an exchange state (see WFS_CMD_CDM_START_EXCHANGE)

WFS_ERR_CDM_NOCASHBOXPRESENT	Cash box amount needed, however teller is not assigned a Cash Box.
WFS_ERR_CDM_AMOUNTNOTINMIXTABLE	A mix table is being used to determine the denomination but the amount specified for the denomination is not in the mix table.

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

Value	Meaning
WFS_EXEE_CDM_CASHUNITERROR	An error occurred while attempting to denominate from the cash unit specified by the event.

Comments None.

5.2 WFS_CMD_CDM_DISPENSE

Description This command performs the dispensing of items to the customer. The command provides the same functionality as the WFS_CMD_CDM_DENOMINATE command plus the additional functionality of dispensing the items. If items of differing currencies are to be dispensed then the currency field must be an array of three ASCII 0x20h characters, the amount must be 0 and the mix number must be WFS_CDM_INDIVIDUAL. However, these restrictions do not apply if a single currency is dispensed with non-currency items, such as coupons.

The WFS_CMD_CDM_DISPENSE command can be used in the following ways:

1. The input parameters to the command are amount, currency and denomination. The mix number is WFS_CDM_INDIVIDUAL. In this case, the denomination is checked for validity and, if valid, is dispensed.
2. The input parameters are amount, currency and mix number. In this case the amount is denominated and, if this succeeds, the items are dispensed.
3. If the amount is 0, but the currency and the denomination are supplied with a mix number of WFS_CDM_INDIVIDUAL the denomination is checked for validity and, if valid, is dispensed.
4. The command will calculate a partial denomination of a given amount and dispense the complete denomination. In this case the input parameters to the command should be currency, amount, mix number and either a partially specified denomination or a minimum amount from the cash box. The cashbox amount may be updated as a result of this command.

When more than one physical cash unit exists for a logical cash unit number, the device selects the actual physical cash unit to use in the dispense operation.

If the *bCashBox* field of the WFSCDMCAPS structure returned by the WFS_INF_CDM_CAPABILITIES command is TRUE then, if the entire denomination cannot be satisfied, a partial denomination will be returned with the remaining amount to be supplied from the Teller’s cash box.

If the device is a Teller CDM, the input parameter *usPosition* can be set to WFS_CDM_POSNULL. If this is the case the *usTellerID* is used to perform the dispense operation to the assigned teller position

The field *bPresent* of the WFSCDMDISPENSE structure determines whether items are actually presented to the user as part of the dispense operation. 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 an intermediate stacker. In the second case it will be necessary to use the WFS_CMD_CDM_PRESENT command to present the items to the user. If *bPresent* is set to FALSE then the *fwPosition* parameter is ignored. If the CDM does not have an intermediate stacker then *bPresent* is ignored.

Input Param

```
LPWFSCDMDISPENSE  lpDispense;
```

```
typedef struct _wfs_cdm_dispense
    {
    USHORT                 usTellerID;
    USHORT                 usMixNumber;
    WORD                   fwPosition;
    BOOL                   bPresent;
    LPWFSCDMDENOMINATION   lpDenomination;
    } WFSCDMDISPENSE, *LPWFSCDMDISPENSE;
```

usTellerID
Identifies the teller. This parameter is ignored if the device is a Self-Service CDM.

usMixNumber
Mix algorithm or house mix table to be used to create a denomination of the supplied amount. If the value is WFS_CDM_INDIVIDUAL, the denomination supplied in the *lpDenomination* field is validated prior to the dispense operation. If it is found to be invalid no alternative denomination will be calculated.

fwPosition
Determines to which side the amount is dispensed. If the device is a Teller CDM this field is ignored and the output position associated with *usTellerID* is used. The value is specified by one of the following values:

Value	Meaning
WFS_CDM_POSNULL	The default configuration information is used. This can be either position dependent or teller dependent.
WFS_CDM_POSLEFT	Present items to left side of device.
WFS_CDM_POSRIGHT	Present items to right side of device.
WFS_CDM_POSCENTER	Present items to center output position.
WFS_CDM_POSTOP	Present items to the top output position.
WFS_CDM_POSBOTTOM	Present items to the bottom output position.
WFS_CDM_POSFRONT	Present items to the front output position.
WFS_CDM_POSREAR	Present items to the rear output position.

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 an intermediate stacker.

lpDenomination
Pointer to a WFSCDMDENOMINATION structure, describing the denominations used for the dispense operation. For the WFSCDMDENOMINATION structure specification see the definition of the command WFS_CMD_CDM_DENOMINATE.

Output Param

```
LPWFSCDMDENOMINATION   lpDenomination;
```

For the WFSCDMDENOMINATION structure specification see the definition of the command WFS_CMD_CDM_DENOMINATE.

The values in this structure report the amount dispensed and the number of items dispensed from each cash unit.

Where mixed currencies are being dispensed the *ulAmount* field in the returned denomination structure will be 0 and the *cCurrency* field will be set to three ASCII 0x20 characters.

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_CDM_INVALIDCURRENCY	There are no cash units in the CDM of the currency specified in the *cCurrency* field of the input parameter.
WFS_ERR_CDM_INVALIDTELLERID	Invalid Teller ID.
WFS_ERR_CDM_CASHUNITERROR	There is a problem with a cash unit. The WFS_EXEE_CDM_CASHUNITERROR execute event is posted with the details.
WFS_ERR_CDM_INVALIDDENOMINATION	The sum of the values for cash box and cash units was greater than the amount specified.

WFS_ERR_CDM_INVALIDMIXNUMBER	Mix algorithm is not known.
WFS_ERR_CDM_NOCURRENCYMIX	Cash units containing two or more different currencies were selected.
WFS_ERR_CDM_NOTDISPENSABLE	The amount is not dispensable by the CDM.
WFS_ERR_CDM_TOOMANYITEMS	The request would require too many items to be dispensed. This error is also generated if *bPresent* is FALSE and sub-dispensing is required.
WFS_ERR_CDM_UNSUPPOSITION	The specified output position is not supported.
WFS_ERR_CDM_SAFEDOOROPEN	The safe door is open.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM is in an exchange state.
WFS_ERR_CDM_NOCASHBOXPRESENT	Cash box amount needed, however teller is not assigned a Cash Box.
WFS_ERR_CDM_AMOUNTNOTINMIXTABLE	A mix table is being used to determine the denomination but the amount specified for the denomination is not in the mix table.
WFS_ERR_CDM_ITEMSNOTTAKEN	Items have not been taken during a sub-dispense operation. This error occurs if a hardware timeout expires.
WFS_ERR_CDM_ITEMSLEFT	Items have been left in the transport or exit slot as a result of a prior Dispense, Present or Recycler Cash-In operation.

If the *bPresent* field of the WFSCDMDISPENSE structure is TRUE, the following error codes can also be returned:

WFS_ERR_CDM_SHUTTERNOTOPEN	The shutter is not open or did not open when it should have. No items presented.
WFS_ERR_CDM_SHUTTEROPEN	The shutter is open when it should be closed. No items presented.
WFS_ERR_CDM_PRERRORNOITEMS	An error occurred while items were being moved to the exit slot - no items are presented.
WFS_ERR_CDM_PRERRORITEMS	An error occurred while items were being moved to the exit slot - at least some of the items have been presented.
WFS_ERR_CDM_PRERRORUNKNOWN	An error occurred while items were being moved to the exit slot - the position of the items is unknown. Intervention may be required to reconcile the cash amount totals.

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_CDM_CASHUNITTHRESHOLD	A threshold condition has been reached in one of the cash units.
WFS_EXEE_CDM_DELAYEDDISPENSE	The dispense operation will be delayed by the specified time.
WFS_EXEE_CDM_STARTDISPENSE	Fired when the delayed dispense operation starts.
WFS_EXEE_CDM_CASHUNITERROR	A cash unit caused an error during a dispense operation.
WFS_SRVE_CDM_ITEMSTAKEN	The user has removed the items presented. If the dispense is not a sub-dispense this event occurs after the completion of the dispense command.
WFS_EXEE_CDM_PARTIALDISPENSE	Indicates that the dispense operation is to be divided into several sub-dispense operations.
WFS_EXEE_CDM_SUBDISPENSEOK	A sub-dispense operation was completed successfully.

WFS_EXEE_CDM_INCOMPLETEDISPENSE	It has not been possible to dispense the entire denomination but part of the denomination has been dispensed, whether on the intermediate stacker or in customer access. The return error code will be WFS_ERR_CDM_NOTDISPENSABLE.
WFS_EXEE_CDM_NOTEERROR	A notes detection error has occurred.

5.3 WFS_CMD_CDM_COUNT

Description This command empties the specified physical cash unit(s). All items dispensed from the cash unit are counted and moved to the specified output location.

The number of items counted can be different from the number of items dispensed in cases where the CDM has the ability to detect this information. If the CDM cannot differentiate between what is dispensed and what is counted then *ulDispensed* will be the same as *ulCounted*.

Upon successful WFS_CMD_CDM_COUNT command execution the physical cash unit(s) *ulCount* field within the WFSCDMPHCU structure is reset.

Input Param
```
LPWFSCDMPHYSICALCU lpPhysicalCU;
```

Pointer to a WFSCDMPHYSICALCU structure:

```
typedef struct _wfs_cdm_physical_cu
{
   BOOL          bEmptyAll;
   WORD          fwPosition;
   LPSTR         lpPhysicalPositionName;
} WFSCDMPHYSICALCU, *LPWFSCDMPHYSICALCU;
```

bEmptyAll
Specifies whether all physical cash units are to be emptied. If this value is TRUE then *lpPhysicalPositionName* is ignored.

fwPosition
A value specifying the location to which items should be moved. The value is set to one of the following values:

Value	Meaning
WFS_CDM_POSNULL	Output location is determined by service provider.
WFS_CDM_POSLEFT	Present items to left side of device.
WFS_CDM_POSRIGHT	Present items to right side of device.
WFS_CDM_POSCENTER	Present items to center output position.
WFS_CDM_POSTOP	Present items to the top output position.
WFS_CDM_POSBOTTOM	Present items to the bottom output position.
WFS_CDM_POSFRONT	Present items to the front output position.
WFS_CDM_POSREAR	Present items to the rear output position.
WFS_CDM_POSREJECT	Reject bin is used as output location.

lpPhysicalPositionName
Identifies which physical cash unit to empty and count. This name is the same as the *lpPhysicalPositionName* in the WFSCDMPHCU structure.

Output Param
```
LPWFSCDMCOUNT lpCount;
```

Pointer to a WFSCDMCOUNT structure:

```
typedef struct _wfs_cdm_count
{
   USHORT                  usNumPhysicalCUs;
   LPWFSCDMCOUNTEDPHYSCU * lppCountedPhysCUs;
} WFSCDMCOUNT, *LPWFSCDMCOUNT;
```

usNumPhysicalCUs
This value indicates the number of physical cash unit structures (WFSCDMCOUNTEDPHYSCU) returned. This value will always be greater than zero.

lppCountedPhysCUs
Pointer to an array of pointers to WFSCDMCOUNTEDPHYSCU structures:

```
typedef struct _wfs_cdm_counted_phys_cu
{
   LPSTR     lpPhysicalPositionName;
   CHAR      cUnitId[5];
   ULONG     ulDispensed;
   ULONG     ulCounted;
   USHORT    usPStatus;
} WFSCDMCOUNTEDPHYSCU, *LPWFSCDMCOUNTEDPHYSCU;
```

lpPhysicalPositionName
Identifies which physical cash unit was emptied and counted. This name is that defined in the *lpPhysicalPositionName* field of the WFSCDMPHCU structure.

cUnitID
Cash unit ID. This is the identifier defined in the *cUnitID* field of the WFSCDMPHCU structure.

ulDispensed
The number of items that were dispensed during the emptying of the cash unit.

ulCounted
The number of items that were counted during the emptying of the cash unit.

usPStatus
Supplies the status of the physical cash unit as one of the following values:

Value	Meaning
WFS_CDM_STATCUOK	The cash unit is in a good state.
WFS_CDM_STATCUFULL	The cash unit is full.
WFS_CDM_STATCUHIGH	The cash unit is almost full (threshold defined by *ulMaximum*).
WFS_CDM_STATCULOW	The cash unit is almost empty (threshold defined by *ulMinimum*).
WFS_CDM_STATCUEMPTY	The cash unit is empty.
WFS_CDM_STATCUINOP	The cash unit is inoperative.
WFS_CDM_STATCUMISSING	The cash unit is missing.
WFS_CDM_STATCUNOVAL	The values of the specified cash unit are not available.
WFS_CDM_STATCUNOREF	There is no reference value available for the notes in this cash unit.
WFS_CDM_STATCUMANIP	The cash unit has been changed when the device was not in the exchange state. This cash unit cannot be dispensed from.

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_CDM_CASHUNITERROR	A cash unit caused a problem. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.
WFS_ERR_CDM_UNSUPPOSITION	The position specified is not supported.
WFS_ERR_CDM_SAFEDOOROPEN	The safe door is open.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM service is in an exchange state.

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_EXEE_CDM_CASHUNITERROR	A cash unit caused an error during the count operation.
WFS_SRVE_CDM_ITEMSTAKEN	The items emptied to the output location have been removed by the user.
WFS_SRVE_CDM_ITEMSPRESENTED	Items have been emptied to the output location. These items may need to be removed from the

output location before the operation can continue.

5.4 WFS_CMD_CDM_PRESENT

Description This command will move items to the exit position for removal by the user.

If a shutter exists, then it will be implicitly controlled during the present operation. The shutter will be closed when the user removes the items or the items are retracted. If *fwPosition* is set to WFS_CDM_POSNULL the position set in the WFS_CMD_CDM_DISPENSE command which caused these items to be dispensed will be used.

When this command successfully completes the items are in customer access.

Input Param LPWORD lpfwPosition

fwPosition
Determines to which position the amount is to be presented. The value is set to one of the following values:

Value	Meaning
WFS_CDM_POSNULL	The default configuration information is used. This can be either position dependent or teller dependent.
WFS_CDM_POSLEFT	Present items to left side of device.
WFS_CDM_POSRIGHT	Present items to right side of device.
WFS_CDM_POSCENTER	Present items to center output position.
WFS_CDM_POSTOP	Present items to the top output position.
WFS_CDM_POSBOTTOM	Present items to the bottom output position.
WFS_CDM_POSFRONT	Present items to the front output position.
WFS_CDM_POSREAR	Present items to the rear output position.

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_CDM_SHUTTERNOTOPEN	The shutter did not open when it should have. No items presented.
WFS_ERR_CDM_SHUTTEROPEN	The shutter is open when it should be closed. No items presented.
WFS_ERR_CDM_NOITEMS	There are no items on the stacker.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM service is in an exchange state.
WFS_ERR_CDM_PRERRORNOITEMS	There was an error during the present operation - no items were presented.
WFS_ERR_CDM_PRERRORITEMS	There was an error during the present operation - at least some of the items were presented.
WFS_ERR_CDM_PRERRORUNKNOWN	There was an error during the present operation - the position of the items is unknown. Intervention may be required to reconcile the cash amount totals.

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_CDM_CASHUNITTHRESHOLD	A threshold condition has been reached in one of the cash units.
WFS_SRVE_CDM_ITEMSTAKEN	The items have been removed by the user. This event is generated after the completion of the present operation.

Comments None.

5.5 WFS_CMD_CDM_REJECT

Description This command will move items from the intermediate stacker and transport to the reject cash unit.

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_CDM_CASHUNITERROR	The reject cash unit caused a problem. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.
WFS_ERR_CDM_NOITEMS	There were no items on the stacker.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM service is in an exchange state.

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_CDM_CASHUNITTHRESHOLD	A reject bin threshold condition has been reached.
WFS_EXEE_CDM_CASHUNITERROR	A cash unit caused an error during the reject operation.

Comments None.

5.6 WFS_CMD_CDM_RETRACT

Description This command will retract items which may have been in customer access. Retracted items will be moved to either a retract cash unit, the reject cash unit, the transport or the intermediate stacker. After the items are retracted the shutter is closed automatically.

The *bRetract* field of the WFSCDMCAPS structure specifies whether or not this command is supported.

Input Param
```
LPWFSCDMRETRACT   lpRetract;

struct _wfs_cdm_retract
   {
   WORD         fwOutputPosition;
   USHORT       usRetractArea;
   USHORT       usIndex;
   } WFSCDMRETRACT, * LPWFSCDMRETRACT;
```

fwOutputPosition
Specifies the output position from which to retract the bills. Possible values are:

Value	Meaning
WFS_CDM_POSNULL	The default configuration information should be used.
WFS_CDM_POSLEFT	Retract items from the left output position
WFS_CDM_POSRIGHT	Retract items from the right output position.
WFS_CDM_POSCENTER	Retract items from the center output position.
WFS_CDM_POSTOP	Retract items from the top output position.
WFS_CDM_POSBOTTOM	Retract items from the bottom output position
WFS_CDM_POSFRONT	Retract items from the front output position
WFS_CDM_POSREAR	Retract items from the rear output position

usRetractArea
This value specifies the area to which the items are to be retracted. Possible values are:

Value	Meaning
WFS_CDM_RA_RETRACT	Retract the items to a retract cash unit.
WFS_CDM_RA_TRANSPORT	Retract the items to the transport.
WFS_CDM_RA_STACKER	Retract the items to the intermediate stacker area.
WFS_CDM_RA_REJECT	Retract the items to a reject cash unit.

usIndex
If *usRetractArea* is set to WFS_CDM_RA_RETRACT this field is the logical retract position inside the container into which the cash is to be retracted. This logical number starts with a value of one (1) for the first retract position and increments by one for each subsequent position. If the container contains several logical retract cash units (of type WFS_CDM_TYPERETRACTCASSETTE in command WFS_INF_CDM_CASH_UNIT_INFO), *usIndex* would be incremented from the first position of the first retract cash unit to the last position of the last retract cash unit defined in WFSCDMCUINFO. The maximum value of *usIndex* is the sum of *ulMaximum* of each retract cash unit. If *usRetractArea* is not set to WFS_CDM_RA_RETRACT the value of this field is ignored.

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_CDM_CASHUNITERROR	The retract cash unit caused a problem. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.
WFS_ERR_CDM_NOITEMS	There were no items to retract.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM is in an exchange state.
WFS_ERR_CDM_SHUTTERNOTCLOSED	The shutter failed to close.
WFS_ERR_CDM_ITEMSTAKEN	Items were present at the output position at the start of the operation, but were removed before the operation was complete - some or all of the items were not retracted.
WFS_ERR_CDM_INVALIDRETRACTPOSITION	The *usIndex* is not supported.
WFS_ERR_CDM_NOTRETRACTAREA	The retract area specified in *usRetractArea* is not supported.

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_CDM_CASHUNITTHRESHOLD	A threshold condition has been reached in the retract or reject cash unit.
WFS_EXEE_CDM_CASHUNITERROR	An error occurred while attempting to retract to the retract or reject cash unit.

Comments None.

5.7 WFS_CMD_CDM_OPEN_SHUTTER

Description This command opens the shutter.

Input Param LPWORD lpfwPosition;

lpfwPosition
Specifies which shutter is to be opened. If the application does not need to specify a shutter, this field can be set to NULL or to WFS_CDM_POSNULL. This field can be set to one of the following values:

Value	Meaning
WFS_CDM_POSNULL	The default configuration information should be used.
WFS_CDM_POSLEFT	Open the shutter at the left output position.
WFS_CDM_POSRIGHT	Open the shutter at the right output position.
WFS_CDM_POSCENTER	Open the shutter at the center output position.
WFS_CDM_POSTOP	Open the shutter at the top output position.
WFS_CDM_POSBOTTOM	Open the shutter at the bottom output position.
WFS_CDM_POSFRONT	Open the shutter at the front output position.
WFS_CDM_POSREAR	Open the shutter at the rear output position.

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_CDM_UNSUPPOSITION	The position specified is not supported.
WFS_ERR_CDM_SHUTTERNOTOPEN	The shutter failed to open.
WFS_ERR_CDM_SHUTTEROPEN	The shutter was already open.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM is in an exchange state.

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

Comments None.

5.8 WFS_CMD_CDM_CLOSE_SHUTTER

Description This command closes the shutter.

Input Param LPWORD lpfwPosition;

lpfwPosition

Specifies which shutter is to be closed. If the application does not need to specify a shutter, this field can be set to NULL or to WFS_CDM_POSNULL. The field should be set to one of the following values:

Value	Meaning
WFS_CDM_POSNULL	The default configuration information should be used.
WFS_CDM_POSLEFT	Close the shutter at the left output position
WFS_CDM_POSRIGHT	Close the shutter at the right output position.
WFS_CDM_POSCENTER	Close the shutter at the center output position.
WFS_CDM_POSTOP	Close the shutter at the top output position.
WFS_CDM_POSBOTTOM	Close the shutter at the bottom output position.
WFS_CDM_POSFRONT	Close the shutter at the front output position.
WFS_CDM_POSREAR	Close the shutter at the rear output position.

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_CDM_UNSUPPOSITION	The position specified is not supported.
WFS_ERR_CDM_SHUTTERCLOSED	The shutter was already closed.
WFS_ERR_CDM_SHUTTERNOTCLOSED	The shutter failed to close.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM is in an exchange state.

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

Comments None.

5.9 WFS_CMD_CDM_SET_TELLER_INFO

Description This command allows the application to set the Teller position and initialise counts for each currency assigned to the Teller. The values set by this command are persistent. This command only applies to Teller CDMs.

Input Param `LPWFSCDMTELLERUPDATE        lpTellerUpdate`

```
typedef struct _wfs_cdm_teller_update
{
USHORT                  usAction;
LPWFSCDMTELLERDETAILS   lpTellerDetails;
} WFSCDMTELLERUPDATE, *LPWFSCDMTELLERUPDATE;
```

usAction
The action to be performed specified as one of the following values:

Value	Meaning
WFS_CDM_CREATE_TELLER	A Teller is to be added.
WFS_CDM_MODIFY_TELLER	Information about an existing Teller is to be modified.
WFS_CDM_DELETE_TELLER	A teller is to be removed.

lpTellerDetails
For a specification of the struct WFSCDMTELLERDETAILS please refer to the WFS_INF_CDM_TELLER_INFO 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_CDM_INVALIDCURRENCY	The specified currency not currently available.
WFS_ERR_CDM_INVALIDTELLERID	The Teller ID is invalid.
WFS_ERR_CDM_UNSUPPOSITION	The position specified is not supported.
WFS_ERR_CDM_EXCHANGEACTIVE	The target teller is current in the middle of an exchange operation.

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_SRVE_CDM_TELLERINFOCHANGED	Teller information has been created, modified or deleted.

Comments None.

5.10 WFS_CMD_CDM_SET_CASH_UNIT_INFO

Description This command is used to adjust information regarding the status and contents of the cash units present in the CDM.

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

This command can only be used to change software counters, thresholds and the application lock. All other fields in the input structure will be ignored.

The following fields of the WFSCDMCASHUNIT structure may be updated by this command:
ulInitialCount
ulCount
ulRejectCount
ulMaximum
ulMinimum
bAppLock

As may the following fields of the WFSCDMPHCU structure:
ulInitialCount
ulCount
ulRejectCount

Any other changes must be performed via an exchange operation.

If the fields *ulCount* and *ulRejectCount* of *lppPhysical* are set to 0 by this command, the application is indicating that it does not wish counts to be maintained for the physical cash units. Counts on the logical cash units will still be maintained and can be used by the application. If the physical counts are set by this command then the logical count will be the sum of the physical counts and any value sent as a logical count will be ignored.

The values set by this command are persistent.

Input Param `LPWFSCDMCUINFO    lpCUInfo;`

The WFSCDMCUINFO structure is specified in the documentation of the WFS_INF_CDM_CASH_UNIT_INFO 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_CDM_INVALIDTELLERID	Invalid Teller ID.
WFS_ERR_CDM_INVALIDCASHUNIT	Invalid cash unit ID.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM is in an exchange state.

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_CDM_CASHUNITTHRESHOLD	A threshold condition has been reached in one of the cash units.
WFS_SRVE_CDM_CASHUNITINFOCHANGED	A cash unit was updated as a result of this command.

Comments None.

5.11 WFS_CMD_CDM_START_EXCHANGE

Description This command puts the CDM in an exchange state, i.e. a state in which cash units can be emptied, replenished, removed or replaced. Other than the updates which can be made via the WFS_CMD_CDM_SET_CASH_UNIT_INFO command (see Section 4.11) all changes to a cash unit must take place while the cash unit is in an exchange state.

In the case of self-configuring cash units which are designed to be replaced with no operator intervention the application should use some trigger to initiate an exchange state when appropriate. For instance, the WFS_SRVE_SAFE_DOOR_OPEN event could trigger the application to call WFS_CMD_CDM_START_EXCHANGE.

This command returns current cash unit information in the form described in the documentation of the WFS_INF_CDM_CASH_UNIT_INFO command. This command will also initiate any physical processes which may be necessary to make the cash units accessible. Before using this command an application should first have ensured that it has exclusive control of the CDM.

This command may return WFS_SUCCESS even if WFS_EXEE_CDM_CASHUNITERROR events are generated. If this command returns WFS_SUCCESS or WFS_ERR_CDM_EXCHANGE_ACTIVE the CDM is in an exchange state.

Once in an exchange state the CDM will only respond to the following commands:

- WFS_CMD_CDM_END_EXCHANGE
- Any **WFS[Async]GetInfo** commands
- **WFSClose** – this will end the exchange state
- WFS_CMD_CDM_SET_MIX_TABLE

Any other commands will result in the error WFS_ERR_CDM_EXCHANGEACTIVE being generated

If an error is returned by this command, the WFS_CMD_CDM_CASH_UNIT_INFO command should be used to determine cash unit information.

If the CDM is part of a compound device together with a CIM (i.e. a cash recycler), exchange operations must be performed separately on each part of the compound device. These operations cannot be performed simultaneously. An exchange state must therefore be initiated on each interface in the following sequence:

CDM
(Lock)
WFS_CMD_CDM_START_EXCHANGE
…exchange action…
WFS_CMD_CDM_END_EXCHANGE
(Unlock)
CIM
(Lock)
WFS_CMD_CIM_START_EXCHANGE
…exchange action…
WFS_CMD_CIM_END_EXCHANGE
(Unlock)

In the case of a recycler, the cash-in cash unit counts are set via the CIM interface and the cash-out cash unit counts are set via the CDM interface. Recycling cash units can be set via either interface. However, if the device has recycle units of multiple currencies and/or denominations, then the CIM interface should be used for exchange operations which affect these units.

Input Param `LPWFSCDMSTARTEX    lpStartEx;`

```
typedef struct _wfs_cdm_start_ex
    {
    WORD          fwExchangeType;
    USHORT        usTellerID;
    USHORT        usCount;
    LPUSHORT      lpusCUNumList;
    } WFSCDMSTARTEX, * LPWFSCDMSTARTEX;
```

fwExchangeType
Specifies the type of cash unit exchange operation. This field should be set to one of the following values:

Value	Meaning
WFS_CDM_EXBYHAND	The cash units will be replenished manually either by filling or emptying the cash unit by hand or by replacing the cash unit.
WFS_CDM_EXTOCASSETTES	Items will be moved from the replenishment container to the bill cash units.

usTellerID
Identifies the teller. If the device is a Self-Service CDM this field is ignored.

usCount
Number of cash units to be exchanged. This is also the size of the array contained in the *lpusCUNumList* field.

lpusCUNumList
Pointer to an array of unsigned shorts containing the logical numbers of the cash units to be exchanged. If an invalid logical number is contained in this list, the command will fail with a WFS_ERR_CDM_CASHUNITERROR error.

Output Param `LPWFSCDMCUINFO    lpCUInfo;`

The WFSCDMCUINFO structure is specified in the documentation of the WFS_INF_CDM_CASH_UNIT_INFO command. This is the complete list of cash units not just the cash units that are to be changed.

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_CDM_INVALIDTELLERID	Invalid Teller ID. This error will never be generated by a Self-Service CDM.
WFS_ERR_CDM_CASHUNITERROR	An error occurred with a cash unit while performing the exchange operation. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.

WFS_ERR_CDM_EXCHANGEACTIVE	The CDM is already in an exchange state.

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

Value	Meaning
WFS_EXEE_CDM_CASHUNITERROR	An error occurred while performing the exchange.
WFS_EXEE_CDM_NOTEERROR	A note detection error has occurred.

Comments None.

5.12 WFS_CMD_CDM_END_EXCHANGE

Description This command will end the exchange state. If any physical action took place as a result of the WFS_CMD_CDM_START_EXCHANGE command then this command will cause the cash units to be returned to their normal physical state. Any necessary device testing will also be initiated. The application can also use this command to update cash unit information in the form described in the documentation of the WFS_INF_CDM_CASH_UNIT_INFO command.

The input parameters to this command may be ignored if the service provider can obtain cash unit information from self-configuring cash units.

If the fields *ulCount*, and *ulRejectCount* of *lppPhysical* are set to 0 by this command, the application is indicating that it does not wish counts to be maintained for the physical cash units. Counts on the logical cash units will still be maintained and can be used by the application. If the physical counts are set by this command then the logical count will be the sum of the physical counts and any value sent as a logical count will be ignored.

If an error occurs during the execution of this command, the application must issue WFS_INF_CDM_CASH_UNIT_INFO to determine the cash unit information.

Even if this command does not return WFS_SUCCESS the exchange state has ended.

The values set by this command are persistent.

Input Param

```
LPWFSCDMCUINFO    lpCUInfo;
```

The WFSCDMCUINFO structure is specified in the documentation for the WFS_INF_CDM_CASH_UNIT_INFO command. This pointer can be NULL if the cash unit information has not changed. Otherwise the parameter must contain the complete list of cash unit structures, not just the ones that have changed.

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_CDM_INVALIDTELLERID	Invalid Teller ID.
WFS_ERR_CDM_CASHUNITERROR	This error is returned if there is a problem with the values set for a cash unit. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.
WFS_ERR_CDM_NOEXCHANGEACTIVE	There is no exchange active.

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

Value	Meaning
WFS_EXEE_CDM_CASHUNITERROR	The values of the cash unit structures are incorrect. The cash unit structure that is incorrect is returned as a parameter on this event.
WFS_SRVE_CDM_CASHUNITINFOCHANGED	A cash unit was changed.
WFS_USRE_CDM_CASHUNITTHRESHOLD	A threshold condition has been reached in one of the cash units.

Comments None.

5.13 WFS_CMD_CDM_OPEN_SAFE_DOOR

Description This command unlocks the safe door or starts the time delay count down prior to unlocking the safe door, if the device supports it. The command completes when the door is unlocked or the timer has started.

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_CDM_EXCHANGEACTIVE	The CDM is in an exchange state.

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

Comments None.

5.14 WFS_CMD_CDM_CALIBRATE_CASH_UNIT

Description This command will cause a vendor dependent sequence of hardware events which will calibrate one or more physical cash units associated with a logical cash unit. This is necessary if a new type of bank note is put into the cash unit as the command enables the CDM to obtain the measures of the new bank notes.

If more than one physical cash unit is associated with the cash unit, it is up to the Service Provider to determine whether all the physical cash units need to be calibrated or if it is sufficient to calibrate for one physical unit and load the data into the others.

This command cannot be used to calibrate cash units which have been locked by the application. A WFS_ERR_CDM_CASHUNITERROR code will be returned and the WFS_EXEE_CDM_CASHUNITERROR event generated.

Input Param `LPWFSCDMCALIBRATE lpCalibrateIn;`

```
typedef struct _wfs_cdm_calibrate
  {
  USHORT                usNumber;
  USHORT                usNumOfBills;
  LPWFSCDMITEMPOSITION *  lpPosition;
  } WFSCDMCALIBRATE, * LPWFSCDMCALIBRATE;
```

usNumber
The logical number of the cash unit.

usNumOfBills
The number of bills to be dispensed during the calibration process.

lpPosition
Specifies where the dispensed items should be moved to. For a description of the WFSCDMITEMPOSITION structure see Section WFS_CMD_CDM_RESET.

Output Param `LPWFSCDMCALIBRATE lpCalibrateOut;`

The WFSCDMCALIBRATE structure is defined in the Input Param section.

usNumber
The logical number of cash unit which has been calibrated

usNumOfBills
Number of items that were actually dispensed during the calibration process. This value may be different from that passed in using the input structure if the cash dispenser always dispenses a default number of bills.

lpPosition
Specifies where the items were moved to during the calibration process.

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_CDM_CASHUNITERROR	A cash unit caused an error. A WFS_EXEE_CDM_CASHUNITERROR event will be sent with the details.
WFS_ERR_CDM_UNSUPPOSITION	The position specified is not valid.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM is in an exchange state.

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

Value	Meaning
WFS_USRE_CDM_CASHUNITTHRESHOLD	A threshold condition has been reached in one of the cash units.
WFS_SRVE_CDM_CASHUNITINFOCHANGED	A cash unit was changed.
WFS_EXEE_CDM_CASHUNITERROR	A cash unit caused an error.
WFS_SRVE_CDM_ITEMSTAKEN	The items were removed.

Comments None.

5.15 WFS_CMD_CDM_SET_MIX_TABLE

Description This command is used to set up the mix table specified by the mix number. Mix tables are persistent and are available to all applications in the system. An amount can be specified as different denominations within the mix table. If the amount is specified more than once the service provider will attempt to denominate or dispense the first amount in the table. If this does not succeed (e.g. because of a cash unit failure) the service provider will attempt to denominate or dispense the next amount in the table. The service provider can only dispense amounts which are explicitly mentioned in the mix table.

If a mix number passed in already exists then the information is overwritten with the new information.

The values set by this command are persistent.

Input Param

```
LPWFSCDMMIXTABLE  lpMixTable;
```

The structure WFSCDMMIXTABLE is defined in the documentation of the command WFS_INF_CDM_MIX_TABLE.

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_CDM_INVALIDMIXNUMBER	The supplied *usMixNumber* is reserved for a predefined mix algorithm.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM is in an exchange state.
WFS_ERR_CDM_INVALIDMIXTABLE	The contents of at least one of the defined rows of the mix table is incorrect.

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

Comments None.

5.16 WFS_CMD_CDM_RESET

Description This command is used by the application to perform a hardware reset which will attempt to return the CDM device to a known good state. This command does not over-ride a lock obtained on another application or service handle, nor can it be performed while the CDM is in the exchange state.

The device will attempt to move any items found anywhere within the device to the cash unit or output position specified in the *lpResetIn* parameter. This may not always be possible because of hardware problems.

If items are found inside the device the WFS_SRVE_CDM_MEDIADETECTED event be generated and will inform the application where the items were actually moved to.

Input Param

```
LPWFSCDMITEMPOSITION    lpResetIn;
```

```
typedef struct _wfs_cdm_itemposition
  {
  USHORT              usNumber;
  LPWFSCDMRETRACT     lpRetractArea;
  WORD                fwOutputPosition;
  } WFSCDMITEMPOSITION * LPWFSCDMITEMPOSITION;
```

usNumber
The *usNumber* of the cash unit to which items found inside the CDM are to be moved. If the items are to be moved to an output position this value is 0 and the output position is defined by *fwOutputPosition*.

lpRetractArea
This field is only used if the cash unit specified by *usNumber* is a retract cash unit. In all other cases this field is set to NULL. For a description of this structure see the WFSCDMRETRACT structure defined in WFS_CMD_CDM_RETRACT

fwOutputPosition
The output position to which items are to be moved. If the *usNumber* is non-zero then this field will be ignored. The value is specified as one of the following values:

Value	Meaning
WFS_CDM_POSNULL	The default configuration
WFS_CDM_POSLEFT	The left output position
WFS_CDM_POSRIGHT	The right output position.
WFS_CDM_POSCENTER	The center output position.
WFS_CDM_POSTOP	The top output position.
WFS_CDM_POSBOTTOM	The bottom output position.
WFS_CDM_POSFRONT	The front output position.
WFS_CDM_POSREAR	The rear output position.

If the application does not wish to specify a cash unit or position it can set this value to NULL. In this case the service provider will determine where to move any items found.

Output Param None.

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

Value	Meaning
WFS_ERR_CDM_CASHUNITERROR	A cash unit caused an error.
WFS_ERR_CDM_UNSUPPOSITION	The position specified is not supported.
WFS_ERR_CDM_INVALIDCASHUNIT	The cash unit number specified is not valid.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM is in the exchange state.

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

Value	Meaning
WFS_USRE_CDM_CASHUNITTHRESHOLD	A threshold condition has been reached in one of the cash units.
WFS_EXEE_CDM_CASUNITERROR	A cash unit caused an error.
WFS_SRVE_CDM_MEDIADETECTED	Media has been found in the device.

Comments None.

5.17 WFS_CMD_CDM_TEST_CASH_UNITS

Description This command is used to test cash units following replenishment. All physical cash units are tested that have a status WFS_CDM_STATCUOK or WFS_CDM_STATCULOW and no application lock. If the hardware is able to do so tests are continued even if an error occurs while testing one of the cash units. The command completes with WFS_SUCCESS if the Service Provider successfully manages to test all of the Cash Units which are low or ok regardless of the outcome of the test. This is the case if all the cash units could be tested and a dispense was possible from at least one of the cash units. WFS_EXEE_CDM CASHUNITERROR events are sent for every cash unit where the test failed. The operation performed to test the cash units is vendor dependent. Items may be dispensed or transported into the reject bin as a result of this command.

This command cannot be used to test cash units which have been locked by the application. A WFS_ERR_CDM_CASHUNITERROR code will be returned and the WFS_EXEE_CDM_CASHUNITERROR event generated.

Input Param LPWFSCDMITEMPOSITION *lpPosition*
Specifies where items dispensed as a result of this command should be moved to. For a description of the WFSCDMITEMPOSITION structure see section WFS_CMD_CDM_RESET.

If a service provider default configuration is to be used this parameter can be NULL.

Output Param `LPWFSCDMCUINFO    lpCUInfo;`

The WFSCDMCUINFO structure is defined in the documentation of the WFS_INF_CDM_CASH_UNIT_INFO command.

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_CDM_CASHUNITERROR	A cash unit caused a problem or the cash unit could not be tested. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.
WFS_ERR_CDM_UNSUPPOSITION	The position specified is not supported.
WFS_ERR_CDM_SHUTTERNOTOPEN	The shutter is not open or did not open when it should have. No items presented.
WFS_ERR_CDM_SHUTTEROPEN	The shutter is open when it should be closed. No items presented.
WFS_ERR_CDM_INVALIDCASHUNIT	The cash unit number specified is not valid.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM service is in an exchange state.
WFS_ERR_CDM_PRERRORNOITEMS	There was an error during the present operation - no items were presented.
WFS_ERR_CDM_PRERRORITEMS	There was an error during the present operation - at least some of the items were presented.
WFS_ERR_CDM_PRERRORUNKNOWN	There was an error during the present operation - the position of the items is unknown. Intervention may be required to reconcile the cash amount totals.

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

Value	Meaning
WFS_USRE_CDM_CASHUNITTHRESHOLD	A threshold condition has been reached in one of the cash units.
WFS_SRVE_CDM_CASHUNITINFOCHANGED	A cash unit was changed.
WFS_EXEE_CDM_CASHUNITERROR	A cash unit has failed the test or a. cash unit could not be tested because it is inoperative, empty or locked.
WFS_SRVE_CDM_ITEMSTAKEN	The items presented have been removed by the user.

6. Events

6.1 WFS_SRVE_CDM_SAFEDOOROPEN

Description	This service event is generated when the safe door has been opened.
Event Param	None.
Comments	None.

6.2 WFS_SRVE_CDM_SAFEDOORCLOSED

Description	This service event is generated when the safe door has been closed.
Event Param	None.
Comments	None.

6.3 WFS_USRE_CDM_CASHUNITTHRESHOLD

Description This user event is generated when a threshold condition has occurred in one of the cash units.

Event Param

```
LPWFSCDMCASHUNIT  lpCashUnit;
```

lpCashUnit
Pointer to WFSCDMCASHUNIT structure, describing the cash unit on which the threshold condition occurred. See *lpCashUnit->usStatus* for the current status. For a description of the WFSCDMCASHUNIT structure, see the definition of the WFS_INF_CDM_CASH_UNIT_INFO command.

Comments None.

6.4 WFS_SRVE_CDM_CASHUNITINFOCHANGED

Description This service event is generated when information about a physical or logical cash unit has changed. For instance, a physical cash unit may have been removed or inserted. This event will also be posted on successful completion of the following commands:

WFS_CMD_CDM_SET_CASH_UNIT_INFO
WFS_CMD_CDM_END_EXCHANGE
WFS_CMD_CDM_CALIBRATE_CASH_UNIT

When a physical cash unit is removed, the status of the physical cash unit becomes WFS_CDM_STATMISSING. If there are no physical cash units of the same logical type remaining the status of the logical type becomes WFS_CDM_STATMISSING.

When a physical cash unit is inserted and this physical cash unit is of an existing logical type, the physical cash unit structure will be updated.

If a physical cash unit of a new logical type is inserted, the *usNumber* of the changed cash unit structure pointed to by *lpCashUnit* is no longer valid. In that case an application should issue a WFS_INF_CDM_CASH_UNIT_INFO command after receiving this event to obtain updated cash unit information.

Event Param

```
LPWFSCDMCASHUNIT  lpCashUnit;
```

lpCashUnit
Pointer to the changed cash unit structure. For a description of the WFSCDMCASHUNIT structure see the definition of the WFS_INF_CDM_CASH_UNIT_INFO command.

Comments None.

6.5 WFS_SRVE_CDM_TELLERINFOCHANGED

Description This service event is generated when the counts assigned to a teller have changed. This event is only returned as a result of a WFS_CMD_CDM_SET_TELLER_INFO command.

Event Param

```
LPUSHORT     lpusTellerID;
```

lpusTellerID
Pointer to an unsigned short holding the ID of the teller whose counts have changed.

Comments None.

6.6 WFS_EXEE_CDM_DELAYEDDISPENSE

Description This execute event is generated if the start of a dispense operation has been delayed.

Event Param

```
LPULONG lpulDelay;
```

lpulDelay
Pointer to the time in milliseconds by which the dispense operation will be delayed.

Comments None.

6.7 WFS_EXEE_CDM_STARTDISPENSE

Description This execute event is generated when a delayed dispense operation begins.

Event Param

```
LPREQUESTID  lpReqID;
```

lpReqID
Pointer to the *RequestID* of the original dispense command.

Comments None.

6.8 WFS_EXEE_CDM_CASHUNITERROR

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

Event Param

```
LPWFSCDMCUERROR   lpCashUnitError;

typedef struct _wfs_cdm_cu_error
    {
    WORD                wFailure;
    LPWFSCDMCASHUNIT    lpCashUnit;
    } WFSCDMCUERROR, * LPWFSCDMCUERROR;
```

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

Value	Meaning
WFS_CDM_CASHUNITEMPTY	Specified cash unit is empty.
WFS_CDM_CASHUNITERROR	Specified cash unit has malfunctioned.
WFS_CDM_CASHUNITFULL	Specified cash unit is full.
WFS_CDM_CASHUNITLOCKED	Specified cash unit is locked.
WFS_CDM_CASHUNITINVALID	Specified cash unit ID is invalid.
WFS_CDM_CASHUNITCONFIG	An attempt has been made to change the settings of a self-configuring cash unit.

lpCashUnit
Pointer to the cash unit structure that caused the problem. The WFSCDMCASHUNIT structure is defined in the documentation of the WFS_INF_CDM_CASH_UNIT_INFO command. It is possible that this pointer may be NULL if the *wFailure* field is WFS_CDM_CASHUNITINVALID.

Comments None.

6.9 WFS_SRVE_CDM_ITEMSTAKEN

Description This service event is generated when items presented to the user have been taken.

Event Param

```
LPWORD  lpfwPosition;
```

The output position from which the items have been removed. Possible values are :

Value	Meaning
WFS_CDM_POSNULL	The default configuration
WFS_CDM_POSLEFT	The left output position
WFS_CDM_POSRIGHT	The right output position.
WFS_CDM_POSCENTER	The center output position.
WFS_CDM_POSTOP	The top output position.
WFS_CDM_POSBOTTOM	The bottom output position
WFS_CDM_POSFRONT	The front output position
WFS_CDM_POSREAR	The rear output position

Comments None.

6.10 WFS_SRVE_CDM_COUNTS_CHANGED

Description This service event is generated if the device is a compound device together with a CIM and the counts in a shared cash unit have changed as a result of a cash-in operation.

Event Param

```
LPWFSCDMCOUNTSCHANGED  lpCountsChanged;

typedef struct _wfs_cdm_counts_changed
{
      USHORT    usCount;
      USHORT *  lpusCUNumList;
} WFSCDMCOUNTSCHANGED, *LPWFSCDMCOUNTSCHANGED;
```

usCount
The size of lpusCUNumList.

lpusCUNumList
A list of the *usNumbers* of the cash units whose counts have changed.

Comments None.

6.11 WFS_EXEE_CDM_PARTIALDISPENSE

Description This execute event is generated when a dispense operation is divided into several sub-dispense operations because the hardware capacity of the CDM is exceeded.

Event Param

```
LPUSHORT     lpusDispNum;
```

lpusDispNum
Specifies the number of sub-dispense operations into which the dispense operation has been divided.

Comments None.

6.12 WFS_EXEE_CDM_SUBDISPENSEOK

Description This execute event is generated when one of the sub-dispense operations into which the dispense operation was divided has finished successfully.

Event Param `LPWFSCDMDENOMINATION   lpDenomination;`

lpDenomination
The WFSCDMDENOMINATION structure is defined in the documentation of the command WFS_CMD_CDM_DENOMINATE. Note that in this case the values in this structure report the amount and number of each denomination dispensed in the sub-dispense operation.

Comments None.

6.13 WFS_EXEE_CDM_INCOMPLETEDISPENSE

Description This execute event is generated when not all of the items specified in a WFS_CMD_CDM_DISPENSE operation could be dispensed. Some of the items have been dispensed. If the device has no intermediate stacker then the bills that were dispensed will be in customer access.

Event Param `LPWFSCDMDENOMINATION   lpDenomination;`

lpDenomination
The WFSCDMDENOMINATION structure is defined in the documentation of the command WFS_CMD_CDM_DENOMINATE. Note that in this case the values in this structure report the amount and number of each denomination that has actually been dispensed.

Comments None.

6.14 WFS_EXEE_CDM_NOTEERROR

Description This execute event specifies the reason for a notes detection error during an exchange or dispense operation.

Event Param `LPUSHORT    lpusReason;`

lpusReason
Specifies the reason for the notes detection error. Possible values are:.

Value	Meaning
WFS_CDM_DOUBLENOTEDETECTED	Double notes have been detected.
WFS_CDM_LONGNOTEDETECTED	A long note has been detected.
WFS_CDM_SKEWEDNOTE	A skewed note has been detected.
WFS_CDM_INCORRECTCOUNT	A bill counting error has occurred.
WFS_CDM_NOTESTOOCLOSE	Notes have been detected as being too close.

Comments None.

6.15 WFS_SRVE_CDM_ITEMSPRESENTED

Description This service event specifies that items have been presented to the user during a Count operation and need to be taken.

Event Param None.

Comments None.

6.16 WFS_SRVE_CDM_MEDIADETECTED

Description This service event is generated if media is detected during a reset (WFS_CMD_CDM_RESET). The parameter on the event informs the application of the position of the media after the reset

	completes. If the device has been unable to successfully move the items found then this parameter will be NULL.
Event Param	LPWFSCDMITEMPOSITION * lpItemPosition; For a description of this parameter see WFS_CMD_CDM_RESET (section 5.16)
Comments	None.

7. Sub-Dispensing Command Flow

“Sub-dispensing” of bills occur when a WFS_CMD_CDM_DISPENSE execute command is issued and the required number of bills to be dispensed exceeds the CDM hardware limit for bills that can be dispensed with a single “hardware level” dispense command. In this situation, the CDM service provider determines the number of “hardware level” dispense commands required and enters what is referred to as a “sub-dispensing” operation until the full amount has been dispensed. Through use of a “sub-dispensing” operation the application is fully removed from “hardware level dependencies” as to how many bills can be dispensed based on hardware vendor design limitations.

The following series of tables illustrate the steps taken on behalf of an end-user, application, XFS service provider, and CDM hardware for sub-dispensing operations: All examples below assume the *bPresent* parameter in the WFS_CMD_CDM_DISPENSE command is set to TRUE.

Sub-Dispensing Is Not Required – Transaction Successful

This table illustrates a successful WFS_CMD_CDM_DISPENSE command where sub-dispensing is not required:

Step	End-User	Application	XFS SP	CDM Hardware
1	User wants to dispense $40.00 USD			
2		WFS_CMD_CDM_DISPENSE command issued.		
3			Determines that a single “hardware level” dispense command can be issued for full dispense request.	
4			“Hardware level” dispense command issued.	
5				items presented.
6		WFS_CMD_CDM_DISPENSE completes successfully.		
7	User takes bills.			
8			WFS_SRVE_CDM_ITEMSSTAKEN event generated.	

Sub-Dispensing Is Required – Command Successful

This table illustrates a successful WFS_CMD_CDM_DISPENSE command where sub-dispensing is required:

Step	End-User	Application	XFS SP	CDM Hardware
1.	User wants to dispense $130 USD in $1 USD denominations.			
2.		WFS_CMD_CDM_DISPENSE command issued.		
3.			Three "hardware level" dispense commands are required. CDM hardware is limited to dispensing 50 bills in any single "hardware level" dispense	
4.			WFS_EXEE_CDM_PARTIALDISPENSE event generated.	
5.			"Hardware level" dispense command issued for $50 USD. command.	
6.				items presented.
7.			WFS_SRVE_CDM_SUBDISPENSEOK event generated.	
8.	User takes bills.			
9.			WFS_SRVE_CDM_ITEMSTAKEN event generated.	
10			"Hardware level" dispense command issued for $50 USD.	
11				items presented.
12			WFS_SRVE_CDM_SUBDISPENSEOK event generated.	
13	User takes bills			
14			WFS_SRVE_CDM_ITEMSTAKEN event generated.	
15			"Hardware level" dispense command issued for $30 USD	
16				items presented.
17			WFS_SRVE_CDM_SUBDISPENSEOK event generated.	
18		WFS_CMD_CDM_DISPENSE completes successfully.		
19	User takes bills			
20			WFS_SRVE_CDM_ITEMSSTAKEN event generated.	

Sub-Dispensing Is Required – Command Unsuccessful

This table illustrates an unsuccessful WFS_CMD_CDM_DISPENSE command where sub-dispensing is required and the end-user does not take the bills during the second "hardware level" dispense, resulting in a timeout condition.

Step	End-User	Application	XFS SP	CDM Hardware
1.	User wants to dispense $130 USD in $1 USD denomination s			
2.		WFS_CMD_CDM_DISPENSE command issued.		
3.			Three "hardware level" dispense commands are required. CDM hardware is limited to dispensing 50 bills in any single "hardware level" dispense command.	
4.			WFS_EXEE_CDM_PARTIALDISPENS E event generated.	
5.			"Hardware level" dispense command issued for $50 USD.	
6.				items presented.
7.			WFS_SRVE_CDM_SUBDISPENSEOK event generated.	
8.	User takes bills.			
9.			WFS_SRVE_CDM_ITEMSTAKEN event generated.	
0.			"Hardware level" dispense command issued for $50 USD.	
11				Items presented.
12			WFS_SRVE_CDM_SUBDISPENSEOK event generated.	
13	User does not take bills.			
14			Timeout occurs waiting on end-user to take bills.	
15		WFS_CMD_CDM_DISPENSE completes with WFS_ERR_CDM_ITEMSNOTTAKEN.		

8. Rules for Cash Unit Exchange

The XFS Start and End Exchange commands should be used by applications to supply the latest information with regards to cash unit replenishment state and content. This guarantees a certain amount of control to an application as to which denominations are stored in which position as well as the general physical state of the logical/physical cash units.

If a cash unit is removed from the CDM outside of the Start/End Exchange operations the status of the physical cash unit should be set to WFS_CDM_STATCUMANIP to indicate to the application that the physical cash unit has been removed and possibly tampered with. While the cash unit has this status the Service Provider should not attempt to use it as part of a Dispense operation. The WFS_CDM_STATCUMANIP status should not change until the next Start/End Exchange operation is performed, even if the cash unit is replaced in its original position.
If all the physical cash units belonging to a logical cash unit are manipulated the parent logical cash unit that the physical cash units belong to should also have its status set to WFS_CDM_STATCUMANIP.

When a cash unit is removed and/or replaced outside of the Start/End Exchange operations the original logical cash unit information such as the values, currency and counts should be preserved in the Cash Unit Info structure reported to the application for accounting purposes until the next Start/End Exchange operations, even if the cash unit physically contains a different denomination.

9. C - Header file

```
/******************************************************************************
*                                                                             *
* xfscdm.h      XFS - Cash Dispenser (CDM) definitions                        *
*                                                                             *
*               Version 3.00 (10/18/00)                                       *
*                                                                             *
******************************************************************************/

#ifndef __INC_XFSCDM__H
#define __INC_XFSCDM__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

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

/* values of WFSCDMCAPS.wClass */

#define     WFS_SERVICE_CLASS_CDM               (3)
#define     WFS_SERVICE_CLASS_VERSION_CDM       0x0003
#define     WFS_SERVICE_CLASS_NAME_CDM          "CDM"

#define     CDM_SERVICE_OFFSET                  (WFS_SERVICE_CLASS_CDM * 100)

/* CDM Info Commands */

#define     WFS_INF_CDM_STATUS                  (CDM_SERVICE_OFFSET + 1)
#define     WFS_INF_CDM_CAPABILITIES            (CDM_SERVICE_OFFSET + 2)
#define     WFS_INF_CDM_CASH_UNIT_INFO          (CDM_SERVICE_OFFSET + 3)
#define     WFS_INF_CDM_TELLER_INFO             (CDM_SERVICE_OFFSET + 4)
#define     WFS_INF_CDM_CURRENCY_EXP            (CDM_SERVICE_OFFSET + 6)
#define     WFS_INF_CDM_MIX_TYPES               (CDM_SERVICE_OFFSET + 7)
#define     WFS_INF_CDM_MIX_TABLE               (CDM_SERVICE_OFFSET + 8)
#define     WFS_INF_CDM_PRESENT_STATUS          (CDM_SERVICE_OFFSET + 9)

/* CDM Execute Commands */

#define     WFS_CMD_CDM_DENOMINATE              (CDM_SERVICE_OFFSET + 1)
#define     WFS_CMD_CDM_DISPENSE                (CDM_SERVICE_OFFSET + 2)
#define     WFS_CMD_CDM_PRESENT                 (CDM_SERVICE_OFFSET + 3)
#define     WFS_CMD_CDM_REJECT                  (CDM_SERVICE_OFFSET + 4)
#define     WFS_CMD_CDM_RETRACT                 (CDM_SERVICE_OFFSET + 5)
#define     WFS_CMD_CDM_OPEN_SHUTTER            (CDM_SERVICE_OFFSET + 7)
#define     WFS_CMD_CDM_CLOSE_SHUTTER           (CDM_SERVICE_OFFSET + 8)
#define     WFS_CMD_CDM_SET_TELLER_INFO         (CDM_SERVICE_OFFSET + 9)
#define     WFS_CMD_CDM_SET_CASH_UNIT_INFO      (CDM_SERVICE_OFFSET + 10)
#define     WFS_CMD_CDM_START_EXCHANGE          (CDM_SERVICE_OFFSET + 11)
#define     WFS_CMD_CDM_END_EXCHANGE            (CDM_SERVICE_OFFSET + 12)
#define     WFS_CMD_CDM_OPEN_SAFE_DOOR          (CDM_SERVICE_OFFSET + 13)
#define     WFS_CMD_CDM_CALIBRATE_CASH_UNIT     (CDM_SERVICE_OFFSET + 15)
#define     WFS_CMD_CDM_SET_MIX_TABLE           (CDM_SERVICE_OFFSET + 20)
#define     WFS_CMD_CDM_RESET                   (CDM_SERVICE_OFFSET + 21)
#define     WFS_CMD_CDM_TEST_CASH_UNITS         (CDM_SERVICE_OFFSET + 22)
#define     WFS_CMD_CDM_COUNT                   (CDM_SERVICE_OFFSET + 23)

/* CDM Messages */

#define     WFS_SRVE_CDM_SAFEDOOROPEN           (CDM_SERVICE_OFFSET + 1)
#define     WFS_SRVE_CDM_SAFEDOORCLOSED         (CDM_SERVICE_OFFSET + 2)
#define     WFS_USRE_CDM_CASHUNITTHRESHOLD      (CDM_SERVICE_OFFSET + 3)
#define     WFS_SRVE_CDM_CASHUNITINFOCHANGED    (CDM_SERVICE_OFFSET + 4)
#define     WFS_SRVE_CDM_TELLERINFOCHANGED      (CDM_SERVICE_OFFSET + 5)
#define     WFS_EXEE_CDM_DELAYEDDISPENSE        (CDM_SERVICE_OFFSET + 6)
#define     WFS_EXEE_CDM_STARTDISPENSE          (CDM_SERVICE_OFFSET + 7)
#define     WFS_EXEE_CDM_CASHUNITERROR          (CDM_SERVICE_OFFSET + 8)
#define     WFS_SRVE_CDM_ITEMSTAKEN             (CDM_SERVICE_OFFSET + 9)
#define     WFS_EXEE_CDM_PARTIALDISPENSE        (CDM_SERVICE_OFFSET + 10)
```

```
#define     WFS_EXEE_CDM_SUBDISPENSEOK          (CDM_SERVICE_OFFSET + 11)
#define     WFS_SRVE_CDM_ITEMSPRESENTED         (CDM_SERVICE_OFFSET + 13)
#define     WFS_SRVE_CDM_COUNTS_CHANGED         (CDM_SERVICE_OFFSET + 14)
#define     WFS_EXEE_CDM_INCOMPLETEDISPENSE     (CDM_SERVICE_OFFSET + 15)
#define     WFS_EXEE_CDM_NOTEERROR              (CDM_SERVICE_OFFSET + 16)
#define     WFS_EXEE_CDM_MEDIADETECTED          (CDM_SERVICE_OFFSET + 17)

/* values of WFSCDMSTATUS.fwDevice */
#define     WFS_CDM_DEVONLINE                   WFS_STAT_DEVONLINE
#define     WFS_CDM_DEVOFFLINE                  WFS_STAT_DEVOFFLINE
#define     WFS_CDM_DEVPOWEROFF                 WFS_STAT_DEVPOWEROFF
#define     WFS_CDM_DEVNODEVICE                 WFS_STAT_DEVNODEVICE
#define     WFS_CDM_DEVHWERROR                  WFS_STAT_DEVHWERROR
#define     WFS_CDM_DEVUSERERROR                WFS_STAT_DEVUSERERROR
#define     WFS_CDM_DEVBUSY                     WFS_STAT_DEVBUSY

/* values of WFSCDMSTATUS.fwSafeDoor */

#define     WFS_CDM_DOORNOTSUPPORTED            (1)
#define     WFS_CDM_DOOROPEN                    (2)
#define     WFS_CDM_DOORCLOSED                  (3)
#define     WFS_CDM_DOORUNKNOWN                 (5)

/* values of WFSCDMSTATUS.fwDispenser */

#define     WFS_CDM_DISPOK                      (0)
#define     WFS_CDM_DISPCUSTATE                 (1)
#define     WFS_CDM_DISPCUSTOP                  (2)
#define     WFS_CDM_DISPCUUNKNOWN               (3)

/* values of WFSCDMSTATUS.fwIntermediateStacker */

#define     WFS_CDM_ISEMPTY                     (0)
#define     WFS_CDM_ISNOTEMPTY                  (1)
#define     WFS_CDM_ISNOTEMPTYCUST              (2)
#define     WFS_CDM_ISNOTEMPTYUNK               (3)
#define     WFS_CDM_ISUNKNOWN                   (4)
#define     WFS_CDM_ISNOTSUPPORTED              (5)


/* values of WFSCDMOUTPOS.fwShutter */

#define     WFS_CDM_SHTCLOSED                   (0)
#define     WFS_CDM_SHTOPEN                     (1)
#define     WFS_CDM_SHTJAMMED                   (2)
#define     WFS_CDM_SHTUNKNOWN                  (3)
#define     WFS_CDM_SHTNOTSUPPORTED             (4)

/* values of WFSCDMOUTPOS.fwPositionStatus */

#define     WFS_CDM_PSEMPTY                     (0)
#define     WFS_CDM_PSNOTEMPTY                  (1)
#define     WFS_CDM_PSUNKNOWN                   (2)
#define     WFS_CDM_PSNOTSUPPORTED              (3)

/* values of WFSCDMOUTPOS.fwTransport */

#define     WFS_CDM_TPOK                        (0)
#define     WFS_CDM_TPINOP                      (1)
#define     WFS_CDM_TPUNKNOWN                   (2)
#define     WFS_CDM_TPNOTSUPPORTED              (3)

/* values of WFSCDMOUTPOS.fwTransportStatus */

#define     WFS_CDM_TPSTATEMPTY                 (0)
#define     WFS_CDM_TPSTATNOTEMPTY              (1)
#define     WFS_CDM_TPSTATNOTEMPTYCUST          (2)
#define     WFS_CDM_TPSTATNOTEMPTY_UNK          (3)
#define     WFS_CDM_TPSTATNOTSUPPORTED          (4)


/* values of WFSCDMCAPS.fwType */

#define     WFS_CDM_TELLERBILL                  (0)
```

```
#define     WFS_CDM_SELFSERVICEBILL             (1)
#define     WFS_CDM_TELLERCOIN                  (2)
#define     WFS_CDM_SELFSERVICECOIN             (3)

/* values of WFSCDMCAPS.fwRetractAreas */
/* values of WFSCDMRETRACT.usRetractArea */

#define     WFS_CDM_RA_RETRACT                  (0x0001)
#define     WFS_CDM_RA_TRANSPORT                (0x0002)
#define     WFS_CDM_RA_STACKER                  (0x0004)
#define     WFS_CDM_RA_REJECT                   (0x0008)
#define     WFS_CDM_RA_NOTSUPP                  (0x0010)

/* values of WFSCDMCAPS.fwRetractTransportActions */
/* values of WFSCDMCAPS.fwRetractStackerActions */

#define     WFS_CDM_PRESENT                     (0x0001)
#define     WFS_CDM_RETRACT                     (0x0002)
#define     WFS_CDM_REJECT                      (0x0004)
#define     WFS_CDM_NOTSUPP                     (0x0008)

/* values of WFSCDMCAPS.fwMoveItems */

#define     WFS_CDM_FROMCU                      (0x0001)
#define     WFS_CDM_TOCU                        (0x0002)
#define     WFS_CDM_TOTRANSPORT                 (0x0004)

/* values of WFSCDMCASHUNIT.usType */

#define     WFS_CDM_TYPENA                      (1)
#define     WFS_CDM_TYPEREJECTCASSETTE          (2)
#define     WFS_CDM_TYPEBILLCASSETTE            (3)
#define     WFS_CDM_TYPECOINCYLINDER            (4)
#define     WFS_CDM_TYPECOINDISPENSER           (5)
#define     WFS_CDM_TYPERETRACTCASSETTE         (6)
#define     WFS_CDM_TYPECOUPON                  (7)
#define     WFS_CDM_TYPEDOCUMENT                (8)
#define     WFS_CDM_TYPEREPCONTAINER           (11)
#define     WFS_CDM_TYPERECYCLING              (12)

/* values of WFSCDMCASHUNIT.usStatus */

#define     WFS_CDM_STATCUOK                    (0)
#define     WFS_CDM_STATCUFULL                  (1)
#define     WFS_CDM_STATCUHIGH                  (2)
#define     WFS_CDM_STATCULOW                   (3)
#define     WFS_CDM_STATCUEMPTY                 (4)
#define     WFS_CDM_STATCUINOP                  (5)
#define     WFS_CDM_STATCUMISSING               (6)
#define     WFS_CDM_STATCUNOVAL                 (7)
#define     WFS_CDM_STATCUNOREF                 (8)
#define     WFS_CDM_STATCUMANIP                 (9)

/* values of WFSCDMMIXTYPE.usMixType */

#define     WFS_CDM_MIXALGORITHM                (1)
#define     WFS_CDM_MIXTABLE                    (2)

/* values of WFSCDMMIXTYPE.usMixNumber */

#define     WFS_CDM_INDIVIDUAL                  (0)

/* values of WFSCDMMIXTYPE.usSubType (predefined mix algorithms) */
#define     WFS_CDM_MIX_MINIMUM_NUMBER_OF_BILLS            (1)
#define     WFS_CDM_MIX_EQUAL_EMPTYING_OF_CASH_UNITS       (2)
#define     WFS_CDM_MIX_MAXIMUM_NUMBER_OF_CASH_UNITS       (3)

/* values of WFSCDMPRESENTSTATUS.wPresentState */

#define     WFS_CDM_PRESENTED                   (1)
#define     WFS_CDM_NOTPRESENTED                (2)
#define     WFS_CDM_UNKNOWN                     (3)

/* values of WFSCDMDISPENSE.fwPosition */
/* values of WFSCDMCAPS.fwPositions */
/* values of WFSCDMOUTPOS.fwPosition */
/* values of WFSCDMTELLERPOS.fwPosition */
/* values of WFSCDMTELLERDETAILS.fwOutputPosition */
/* values of WFSCDMPHYSICALCU.fwPosition */

#define     WFS_CDM_POSNULL                     (0x0000)
#define     WFS_CDM_POSLEFT                     (0x0001)
#define     WFS_CDM_POSRIGHT                    (0x0002)
#define     WFS_CDM_POSCENTER                   (0x0004)
#define     WFS_CDM_POSTOP                      (0x0040)
#define     WFS_CDM_POSBOTTOM                   (0x0080)
#define     WFS_CDM_POSREJECT                   (0x0100)
#define     WFS_CDM_POSFRONT                    (0x0800)
#define     WFS_CDM_POSREAR                     (0x1000)

/* values of WFSCDMTELLERDETAILS.ulInputPosition */
#define     WFS_CDM_POSINLEFT                   (0x0001)
#define     WFS_CDM_POSINRIGHT                  (0x0002)
#define     WFS_CDM_POSINCENTER                 (0x0004)
#define     WFS_CDM_POSINTOP                    (0x0008)
#define     WFS_CDM_POSINBOTTOM                 (0x0010)
#define     WFS_CDM_POSINFRONT                  (0x0020)
#define     WFS_CDM_POSINREAR                   (0x0040)

/* values of fwExchangeType */
#define     WFS_CDM_EXBYHAND                    (0x0001)
#define     WFS_CDM_EXTOCASSETTES               (0x0002)


/* values of WFSCDMTELLERUPDATE.usAction */

#define     WFS_CDM_CREATE_TELLER               (1)
#define     WFS_CDM_MODIFY_TELLER               (2)
#define     WFS_CDM_DELETE_TELLER               (3)


/* values of WFSCDMCUERROR.wFailure */

#define     WFS_CDM_CASHUNITEMPTY               (1)
#define     WFS_CDM_CASHUNITERROR               (2)
#define     WFS_CDM_CASHUNITFULL                (4)
#define     WFS_CDM_CASHUNITLOCKED              (5)
#define     WFS_CDM_CASHUNITINVALID             (6)
#define     WFS_CDM_CASHUNITCONFIG              (7)


/* values of lpusReason in WFS_EXEE_CDM_NOTESERROR */

#define     WFS_CDM_DOUBLENOTEDETECTED          (1)
#define     WFS_CDM_LONGNOTEDETECTED            (2)
#define     WFS_CDM_SKEWEDNOTE                  (3)
#define     WFS_CDM_INCORRECTCOUNT              (4)
#define     WFS_CDM_NOTESTOOCLOSE               (5)


/* WOSA/XFS CDM Errors */

#define WFS_ERR_CDM_INVALIDCURRENCY         (-(CDM_SERVICE_OFFSET + 0))
#define WFS_ERR_CDM_INVALIDTELLERID         (-(CDM_SERVICE_OFFSET + 1))
#define WFS_ERR_CDM_CASHUNITERROR           (-(CDM_SERVICE_OFFSET + 2))
#define WFS_ERR_CDM_INVALIDDENOMINATION     (-(CDM_SERVICE_OFFSET + 3))
#define WFS_ERR_CDM_INVALIDMIXNUMBER        (-(CDM_SERVICE_OFFSET + 4))
#define WFS_ERR_CDM_NOCURRENCYMIX           (-(CDM_SERVICE_OFFSET + 5))
#define WFS_ERR_CDM_NOTDISPENSABLE          (-(CDM_SERVICE_OFFSET + 6))
#define WFS_ERR_CDM_TOOMANYITEMS            (-(CDM_SERVICE_OFFSET + 7))
#define WFS_ERR_CDM_UNSUPPOSITION           (-(CDM_SERVICE_OFFSET + 8))
#define WFS_ERR_CDM_SAFEDOOROPEN            (-(CDM_SERVICE_OFFSET + 10))
#define WFS_ERR_CDM_SHUTTERNOTOPEN          (-(CDM_SERVICE_OFFSET + 12))
#define WFS_ERR_CDM_SHUTTEROPEN             (-(CDM_SERVICE_OFFSET + 13))
#define WFS_ERR_CDM_SHUTTERCLOSED           (-(CDM_SERVICE_OFFSET + 14))
#define WFS_ERR_CDM_INVALIDCASHUNIT         (-(CDM_SERVICE_OFFSET + 15))
#define WFS_ERR_CDM_NOITEMS                 (-(CDM_SERVICE_OFFSET + 16))
#define WFS_ERR_CDM_EXCHANGEACTIVE          (-(CDM_SERVICE_OFFSET + 17))
```

```
#define WFS_ERR_CDM_NOEXCHANGEACTIVE        (-(CDM_SERVICE_OFFSET + 18))
#define WFS_ERR_CDM_SHUTTERNOTCLOSED        (-(CDM_SERVICE_OFFSET + 19))
#define WFS_ERR_CDM_PRERRORNOITEMS          (-(CDM_SERVICE_OFFSET + 20))
#define WFS_ERR_CDM_PRERRORITEMS            (-(CDM_SERVICE_OFFSET + 21))
#define WFS_ERR_CDM_PRERRORUNKNOWN          (-(CDM_SERVICE_OFFSET + 22))
#define WFS_ERR_CDM_ITEMSTAKEN              (-(CDM_SERVICE_OFFSET + 23))
#define WFS_ERR_CDM_INVALIDMIXTABLE         (-(CDM_SERVICE_OFFSET + 27))
#define WFS_ERR_CDM_OUTPUTPOS_NOT_EMPTY     (-(CDM_SERVICE_OFFSET + 28))
#define WFS_ERR_CDM_INVALIDRETRACTPOSITION  (-(CDM_SERVICE_OFFSET + 29))
#define WFS_ERR_CDM_NOTRETRACTAREA          (-(CDM_SERVICE_OFFSET + 30))
#define WFS_ERR_CDM_NOCASHBOXPRESENT        (-(CDM_SERVICE_OFFSET + 33))
#define WFS_ERR_CDM_AMOUNTNOTINMIXTABLE     (-(CDM_SERVICE_OFFSET + 34))
#define WFS_ERR_CDM_ITEMSNOTTAKEN           (-(CDM_SERVICE_OFFSET + 35))
#define WFS_ERR_CDM_ITEMSLEFT               (-(CDM_SERVICE_OFFSET + 36))


/*=================================================================*/
/* CDM Info Command Structures */
/*=================================================================*/

typedef struct _wfs_cdm_position
{
    WORD            fwPosition;
    WORD            fwShutter;
    WORD            fwPositionStatus;
    WORD            fwTransport;
    WORD            fwTransportStatus;
} WFSCDMOUTPOS, * LPWFSCDMOUTPOS;

typedef struct _wfs_cdm_status
{
    WORD             fwDevice;
    WORD             fwSafeDoor;
    WORD             fwDispenser;
    WORD             fwIntermediateStacker;
    LPWFSCDMOUTPOS * lppPositions;
    LPSTR            lpszExtra;
} WFSCDMSTATUS, * LPWFSCDMSTATUS;

typedef struct _wfs_cdm_caps
{
    WORD            wClass;
    WORD            fwType;
    WORD            wMaxDispenseItems;
    BOOL            bCompound;
    BOOL            bShutter;
    BOOL            bShutterControl;
    WORD            fwRetractAreas;
    WORD            fwRetractTransportActions;
    WORD            fwRetractStackerActions;
    BOOL            bSafeDoor;
    BOOL            bCashBox;
    BOOL            bIntermediateStacker;
    BOOL            bItemsTakenSensor;
    WORD            fwPositions;
    WORD            fwMoveItems;
    WORD            fwExchangeType;
    LPSTR           lpszExtra;
} WFSCDMCAPS, * LPWFSCDMCAPS;

typedef struct _wfs_cdm_physicalcu
{
    LPSTR           lpPhysicalPositionName;
    CHAR            cUnitID[5];
    ULONG           ulInitialCount;
    ULONG           ulCount;
    ULONG           ulRejectCount;
    ULONG           ulMaximum;
    USHORT          usPStatus;
    BOOL            bHardwareSensor;
} WFSCDMPHCU, * LPWFSCDMPHCU;

typedef struct _wfs_cdm_cashunit
{
    USHORT          usNumber;
    USHORT          usType;
    LPSTR           lpszCashUnitName;
    CHAR            cUnitID[5];
    CHAR            cCurrencyID[3];
    ULONG           ulValues;
    ULONG           ulInitialCount;
    ULONG           ulCount;
    ULONG           ulRejectCount;
    ULONG           ulMinimum;
    ULONG           ulMaximum;
    BOOL            bAppLock;
    USHORT          usStatus;
    USHORT          usNumPhysicalCUs;
    LPWFSCDMPHCU   *lppPhysical;
} WFSCDMCASHUNIT, * LPWFSCDMCASHUNIT;

typedef struct _wfs_cdm_cu_info
{
    USHORT          usTellerID;
    USHORT          usCount;
    LPWFSCDMCASHUNIT *lppList;
} WFSCDMCUINFO, * LPWFSCDMCUINFO;

typedef struct _wfs_cdm_teller_info
{
    USHORT          usTellerID;
    CHAR            cCurrencyID[3];
} WFSCDMTELLERINFO, * LPWFSCDMTELLERINFO;

typedef struct _wfs_cdm_teller_totals
{
   char             cCurrencyID[3];
   ULONG            ulItemsReceived;
   ULONG            ulItemsDispensed;
   ULONG            ulCoinsReceived;
   ULONG            ulCoinsDispensed;
   ULONG            ulCashBoxReceived;
   ULONG            ulCashBoxDispensed;
} WFSCDMTELLERTOTALS, * LPWFSCDMTELLERTOTALS;

typedef struct _wfs_cdm_teller_details
{
    USHORT                usTellerID;
    ULONG                 ulInputPosition;
    WORD                  fwOutputPosition;
    LPWFSCDMTELLERTOTALS *lppTellerTotals;
} WFSCDMTELLERDETAILS, * LPWFSCDMTELLERDETAILS;


typedef struct _wfs_cdm_currency_exp
{
    CHAR            cCurrencyID[3];
    SHORT           sExponent;
} WFSCDMCURRENCYEXP, * LPWFSCDMCURRENCYEXP;

typedef struct _wfs_cdm_mix_type
{
    USHORT          usMixNumber;
    USHORT          usMixType;
    USHORT          usSubType;
    LPSTR           lpszName;
} WFSCDMMIXTYPE, * LPWFSCDMMIXTYPE;

typedef struct _wfs_cdm_mix_row
{
    ULONG           ulAmount;
    LPUSHORT        lpusMixture;
} WFSCDMMIXROW, * LPWFSCDMMIXROW;

typedef struct _wfs_cdm_mix_table
{
    USHORT          usMixNumber;
    LPSTR           lpszName;
    USHORT          usRows;
    USHORT          usCols;
    LPULONG         lpulMixHeader;
    LPWFSCDMMIXROW  *lppMixRows;
} WFSCDMMIXTABLE, * LPWFSCDMMIXTABLE;

typedef struct _wfs_cdm_denomination
{
    CHAR            cCurrencyID[3];
    ULONG           ulAmount;
    USHORT          usCount;
    LPULONG         lpulValues;
    ULONG           ulCashBox;
} WFSCDMDENOMINATION, * LPWFSCDMDENOMINATION;

typedef struct _wfs_cdm_present_status
{
    LPWFSCDMDENOMINATION  lpDenomination;
    WORD                  wPresentState;
    LPSTR                 lpszExtra;
} WFSCDMPRESENTSTATUS, * LPWFSCDMPRESENTSTATUS;


/*=================================================================*/
/* CDM Execute Command Structures */
/*=================================================================*/

typedef struct _wfs_cdm_denominate
{
    USHORT                usTellerID;
    USHORT                usMixNumber;
    LPWFSCDMDENOMINATION  lpDenomination;
} WFSCDMDENOMINATE, * LPWFSCDMDENOMINATE;

typedef struct _wfs_cdm_dispense
{
    USHORT                usTellerID;
    USHORT                usMixNumber;
    WORD                  fwPosition;
    BOOL                  bPresent;
    LPWFSCDMDENOMINATION  lpDenomination;
} WFSCDMDISPENSE, * LPWFSCDMDISPENSE;

typedef struct _wfs_cdm_physical_cu
{
    BOOL      bEmptyAll;
    WORD      fwPosition;
    LPSTR     lpPhysicalPositionName;
} WFSCDMPHYSICALCU, *LPWFSCDMPHYSICALCU;

typedef struct _wfs_cdm_counted_phys_cu
{
    LPSTR        lpPhysicalPositionName;
    CHAR         cUnitId[5];
    ULONG        ulDispensed;
    ULONG        ulCounted;
    USHORT       usPStatus;
} WFSCDMCOUNTEDPHYSCU, *LPWFSCDMCOUNTEDPHYSCU;

typedef struct _wfs_cdm_count
{
    USHORT                  usNumPhysicalCUs;
    LPWFSCDMCOUNTEDPHYSCU  *lppCountedPhysCUs;
} WFSCDMCOUNT, *LPWFSCDMCOUNT;


typedef struct _wfs_cdm_retract
{
    WORD                  fwOutputPosition;
    USHORT                usRetractArea;
    USHORT                usIndex;
} WFSCDMRETRACT, * LPWFSCDMRETRACT;

typedef struct _wfs_cdm_teller_update
```

```
{
    USHORT                usAction;
    LPWFSCDMTELLERDETAILS lpTellerDetails;
} WFSCDMTELLERUPDATE, * LPWFSCDMTELLERUPDATE;


typedef struct _wfs_cdm_start_ex
{
    WORD             fwExchangeType;
    USHORT           usTellerID;
    USHORT           usCount;
    LPUSHORT         lpusCUNumList;
} WFSCDMSTARTEX, * LPWFSCDMSTARTEX;

typedef struct _wfs_cdm_itemposition
{
    USHORT             usNumber;
    LPWFSCDMRETRACT    lpRetractArea;
    WORD               fwOutputPosition;
} WFSCDMITEMPOSITION, * LPWFSCDMITEMPOSITION;

typedef struct _wfs_cdm_calibrate
{
    USHORT                   usNumber;
    USHORT                   usNumOfBills;
    LPWFSCDMITEMPOSITION    *lpPosition;
} WFSCDMCALIBRATE, * LPWFSCDMCALIBRATE;


/*=================================================================*/
/* CDM Message Structures */
/*=================================================================*/

typedef struct _wfs_cdm_cu_error
{
    WORD              wFailure;
    LPWFSCDMCASHUNIT  lpCashUnit;
} WFSCDMCUERROR, * LPWFSCDMCUERROR;

typedef struct _wfs_cdm_counts_changed
{
    USHORT            usCount;
    USHORT           *lpusCUNumList;
} WFSCDMCOUNTSCHANGED, * LPWFSCDMCOUNTSCHANGED;


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

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

#endif  /* __INC_XFSCDM__H */
```