CEN

WORKSHOP

AGREEMENT

CWA 16926-64

January 2023

ICS 35.200; 35.240.15; 35.240.40

English version

Extensions for Financial Services (XFS) interface specification Release 3.50 - Part 64: Cash Dispenser Module Class Interface - Programmer's Reference - Migration from Version 3.40 (CWA 16926:2020) to Version 3.50 (this CWA)

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

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

Ref. No.:CWA 16926-64:2023 E

Table of Contents

European Foreword

This CEN Workshop Agreement has been developed in accordance with the CEN-CENELEC Guide 29 “CEN/CENELEC Workshop Agreements – The way to rapid consensus” and with the relevant provisions of CEN/CENELEC Internal Regulations – Part 2. It was approved by a Workshop of representatives of interested parties on 2022-11-08, the constitution of which was supported by CEN following several public calls for participation, the first of which was made on 1998-06-24. However, this CEN Workshop Agreement does not necessarily include all relevant stakeholders.

The final text of this CEN Workshop Agreement was provided to CEN for publication on 2022-11-18.

The following organizations and individuals developed and approved this CEN Workshop Agreement:

- AURIGA SPA
- CIMA SPA
- DIEBOLD NIXDORF SYSTEMS GMBH
- FIS BANKING SOLUTIONS UK LTD (OTS)
- FUJITSU TECHNOLOGY SOLUTIONS
- GLORY LTD
- GRG BANKING EQUIPMENT HK CO LTD
- HITACHI CHANNEL SOLUTIONS CORP
- HYOSUNG TNS INC
- JIANGSU GUOGUANG ELECTRONIC INFORMATION TECHNOLOGY
- KAL
- KEBA HANDOVER AUTOMATION GMBH
- NCR FSG
- NEXUS SOFTWARE
- OBERTHUR CASH PROTECTION
- OKI ELECTRIC INDUSTRY SHENZHEN
- SALZBURGER BANKEN SOFTWARE
- SECURE INNOVATION
- SIGMA SPA

It is possible that some elements of this CEN/CWA may be subject to patent rights. The CEN-CENELEC policy on patent rights is set out in CEN-CENELEC Guide 8 “Guidelines for Implementation of the Common IPR Policy on Patents (and other statutory intellectual property rights based on inventions)”. CEN shall not be held responsible for identifying any or all such patent rights.

The Workshop participants have made every effort to ensure the reliability and accuracy of the technical and non-technical content of CWA 16926-5, but this does not guarantee, either explicitly or implicitly, its correctness. Users of CWA 16926-5 should be aware that neither the Workshop participants, nor CEN can be held liable for damages

or losses of any kind whatsoever which may arise from its application. Users of CWA 16926-5 do so on their own responsibility and at their own risk.

The CWA is published as a multi-part document, consisting of:

Part 1: Application Programming Interface (API) - Service Provider Interface (SPI) - Programmer's Reference

Part 2: Service Classes Definition - Programmer's Reference

Part 3: Printer and Scanning Device Class Interface - Programmer's Reference

Part 4: Identification Card Device Class Interface - Programmer's Reference

Part 5: Cash Dispenser Device Class Interface - Programmer's Reference

Part 6: PIN Keypad Device Class Interface - Programmer's Reference

Part 7: Check Reader/Scanner Device Class Interface - Programmer's Reference

Part 8: Depository Device Class Interface - Programmer's Reference

Part 9: Text Terminal Unit Device Class Interface - Programmer's Reference

Part 10: Sensors and Indicators Unit Device Class Interface - Programmer's Reference

Part 11: Vendor Dependent Mode Device Class Interface - Programmer's Reference

Part 12: Camera Device Class Interface - Programmer's Reference

Part 13: Alarm Device Class Interface - Programmer's Reference

Part 14: Card Embossing Unit Device Class Interface - Programmer's Reference

Part 15: Cash-In Module Device Class Interface - Programmer's Reference

Part 16: Card Dispenser Device Class Interface - Programmer's Reference

Part 17: Barcode Reader Device Class Interface - Programmer's Reference

Part 18: Item Processing Module Device Class Interface - Programmer's Reference

Part 19: Biometrics Device Class Interface - Programmer's Reference

Parts 20 - 28: Reserved for future use.

Parts 29 through 47 constitute an optional addendum to this CWA. They define the integration between the SNMP standard and the set of status and statistical information exported by the Service Providers.

Part 29: XFS MIB Architecture and SNMP Extensions - Programmer’s Reference

Part 30: XFS MIB Device Specific Definitions - Printer Device Class

Part 31: XFS MIB Device Specific Definitions - Identification Card Device Class

Part 32: XFS MIB Device Specific Definitions - Cash Dispenser Device Class

Part 33: XFS MIB Device Specific Definitions - PIN Keypad Device Class

Part 34: XFS MIB Device Specific Definitions - Check Reader/Scanner Device Class

Part 35: XFS MIB Device Specific Definitions - Depository Device Class

Part 36: XFS MIB Device Specific Definitions - Text Terminal Unit Device Class

Part 37: XFS MIB Device Specific Definitions - Sensors and Indicators Unit Device Class

Part 38: XFS MIB Device Specific Definitions - Camera Device Class

Part 39: XFS MIB Device Specific Definitions - Alarm Device Class

Part 40: XFS MIB Device Specific Definitions - Card Embossing Unit Class

Part 41: XFS MIB Device Specific Definitions - Cash-In Module Device Class

Part 42: Reserved for future use.

Part 43: XFS MIB Device Specific Definitions - Vendor Dependent Mode Device Class

Part 44: XFS MIB Application Management

Part 45: XFS MIB Device Specific Definitions - Card Dispenser Device Class

Part 46: XFS MIB Device Specific Definitions - Barcode Reader Device Class

Part 47: XFS MIB Device Specific Definitions - Item Processing Module Device Class

Part 48: XFS MIB Device Specific Definitions - Biometrics Device Class

Parts 49 - 60 are reserved for future use.

Part 61: Application Programming Interface (API) - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Service Provider Interface (SPI) - Programmer's Reference

Part 62: Printer and Scanning Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 63: Identification Card Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 65: PIN Keypad Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 67: Depository Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 71: Camera Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 72: Alarm Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 74: Cash-In Module Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 75: Card Dispenser Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 76: Barcode Reader Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 77: Item Processing Module Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 78: Biometric Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

In addition to these Programmer's Reference specifications, the reader of this CWA is also referred to a complementary document, called Release Notes. The Release Notes contain clarifications and explanations on the CWA specifications, which are not requiring functional changes. The current version of the Release Notes is available online from: https://www.cencenelec.eu/areas-of-work/cen-sectors/digital-society-cen/cwa-download-area/.

The information in this document represents the Workshop's current views on the issues discussed as of the date of publication. It is provided for informational purposes only and is subject to change without notice. CEN makes no warranty, express or implied, with respect to this document.

Revision History:

3.00	October 18, 2000	Initial Release.
3.10	November 29, 2007	For a description of changes from version 3.00 to version 3.10 see the CDM 3.10 Migration document.
3.20	March 2, 2011	For a description of changes from version 3.10 to version 3.20 see the CDM 3.20 Migration document.
3.30	March 19, 2015	For a description of changes from version 3.20 to version 3.30 see the CDM 3.30 Migration document.
3.40	December 06, 2019	For a description of changes from version 3.30 to version 3.40 see the CDM 3.40 Migration document.
3.50	November 18, 2022	For a description of changes from version 3.40 to version 3.50 see the CDM 3.50 Migration document.

1. Introduction

1.1 Background to Release 3.50

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

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

Release 3.50 of the XFS specification is based on a C API and is delivered with the continued promise for the protection of technical investment for existing applications. This release of the specification extends the functionality and capabilities of the existing devices covered by the specification:

- Addition of E2E security
- PIN Password Entry

1.2 XFS Service-Specific Programming

The service classes are defined by their service-specific commands and the associated data structures, error codes, messages, etc. These commands are used to request functions that are specific to one or more classes of Service Providers, but not all of them, and therefore are not included in the common API for basic or administration functions.

When a service-specific command is common among two or more classes of Service Providers, the syntax of the command is as similar as possible across all services, since a major objective of XFS is to standardize function codes and structures for the broadest variety of services. For example, using the **WFSExecute** function, the commands to read data from various services are as similar as possible to each other in their syntax and data structures.

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

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

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

The requested capability is defined for the class of Service Providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability ***is*** considered to be fundamental to the service. In this case, a WFS_ERR_UNSUPP_COMMAND error for Execute commands or WFS_ERR_UNSUPP_CATEGORY error for Info commands is returned to the calling application. An example would be a request from an application to a cash dispenser to retract items where the dispenser hardware does not have that capability; the Service Provider recognizes the command but, since the cash dispenser it is managing is unable to fulfil the request, returns this error.

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

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

2. Cash Dispensers

This specification describes the functionality of an 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.

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.

The CIM interface can be used for all exchange operations on recycle devices, and the CIM interface should be used if the device has recycle units of multiple currencies and/or denominations (including multiple note identifiers associated with the same denomination).

The event WFS_SRVE_CDM_COUNTS_CHANGED will be posted if an operation on the CIM interface affects the 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_END
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_REPLENISH
WFS_CMD_CIM_CASH_UNIT_COUNT

3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (-SPI), Programmer's Reference, Revision 3.~~40~~50
2. ISO 4217 at ~~http://www.iso.org~~http://www.iso.org
3. XFS Cash-In Module Device Class Interface, Programmer's Reference, Revision 3.~~40~~50

4. Note Classification

Notes are classified by the XFS CDM specification according to the following definitions:

1. Level 1 – Note is not recognized.
2. Level 2 – Recognized counterfeit note.
3. Level 3 – Suspected counterfeit note.
4. Level 4 – Recognized note that is identified as genuine. This includes notes which are fit or unfit for recycling.

This definition allows support for legislative note handling standards that may exist in various countries and economic regions. Local requirements or device capability may dictate that notes are not classified as level 2 and level 3 and therefore counterfeit or suspect notes would be classified as level 1; the P6 string reported by WFS_INF_CIM_CAPABILITIES *lpszExtra* reports whether notes are classified into all 4 levels.

The above classification levels can be used to support note handling functionality which includes:

1. The ability to remove counterfeit notes from circulation.
2. Reporting of unrecognized, recognized counterfeit and suspected counterfeit notes.
3. Creating and reporting of note signatures in order to allow back-tracing of notes.

A note's classification can be changed based on the note's serial number, currency and value by specifying a blacklist or classification list. A blacklist reclassifies a matching note as level 2, whereas a classification list can be used to re-classify a matching note to a lower level, including classifying a genuine note as unfit for dispensing. Once reclassified, the note will be automatically handled according to the local country specific note handling standard or legislation. Any reclassification will result in the normal events and behavior, for example a WFS_EXEE_CDM_INFO_AVAILABLE event will reflect the note's reclassification. Reclassification can be used to make dynamic changes to note handling procedures without a software upgrade, enabling functionality such as taking older notes out of circulation or handling of counterfeit notes on a local basis.

Reclassification cannot be used to change a note's classification to a higher level, for example, a note recognized as counterfeit by the device cannot be reclassified as genuine. In addition, it is not possible to re-classify a level 2 note as level 1. No particular use case has been identified for reclassifying Level 3 and 4 notes as level 1, but there is no reason to restrict this reclassification.

Blacklists can be specified using WFS_CMD_CDM_SET_BLACKLIST and retrieved using WFS_INF_CDM_GET_BLACKLIST. Classification lists can be specified using WFS_CMD_CDM_SET_CLASSIFICATION_LIST and retrieved using WFS_INF_CDM_GET_CLASSIFICATION_LIST. A classification list is a superset of the blacklist; any items specified as level 2 in the classification list are considered part of the blacklist. However, it is not recommended that both sets of commands are used by a single application, as it may lead to overlap and confusion.

The blacklist or classification list functionality can use a mask to specify serial numbers. The mask is defined as follows: A '?' character (0x003F) is the wildcard used to match a single Unicode character, and a '*' character (0x002A) is the wildcard used to match one or more Unicode characters.

For example, "S8H9??16?4" would represent a match for the serial numbers "S8H9231654" and "S8H9761684". A mask of "HD90*2" would be used in order to match serial numbers that begin with "HD90" and end with "2", for example "HD9028882", "HD9083276112". Note that the mask can only use one asterisk, and if a real character is required then it must be preceded by a backslash, for example: '\\' for a backslash, '*' for an asterisk or '\?' for a question mark.

Note that this flexibility means that it is possible to overlap definitions, for example "HD90*" and "HD902*" would both match on the serial number HD9028882".

5. Info Commands

5.1 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;
     DWORD                     dwGuidLights[WFS_CDM_GUIDLIGHTS_SIZE];
     WORD                      wDevicePosition;
     USHORT                    usPowerSaveRecoveryTime;
     WORD                      wAntiFraudModule;
     } WFSCDMSTATUS, *LPWFSCDMSTATUS;
```

fwDevice
Supplies the state of the CDM. However, an *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).
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.
WFS_CDM_DEVFRAUDATTEMPT	The device is present but is inoperable because it has detected a fraud attempt.
WFS_CDM_DEVPOTENTIALFRAUD	The device has detected a potential fraud attempt and is capable of remaining in service. In this case the application should make the decision as to whether to take the device offline.

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 safe 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 safe 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	One or more of the cash units is in a low, empty, inoperative or manipulated 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. No items can be dispensed because all of the cash units are in an empty, inoperative or manipulated condition. This state may also occur when a reject/retract cash unit is full or no reject/retract cash unit is present, or when an application lock is set on every cash unit which can be locked.
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;
     WORD                      fwJammedShutterPosition;
     } 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 operational and is closed.
WFS_CDM_SHTOPEN	The shutter is operational and is open.
WFS_CDM_SHTJAMMED	The shutter is jammed and is not operational. The field *fwJammedShutterPosition* provides the positional state of the shutter.
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. The transport is defined as any area leading to or from the position:

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.

fwJammedShutterPosition
Returns information regarding the position of the jammed shutter. The possible values of this field are:

Value	Meaning
WFS_CDM_SHUTTERPOS_NOTSUPPORTED	The physical device has no shutter or the reporting of the position of a jammed shutter is not supported.
WFS_CDM_SHUTTERPOS_NOTJAMMED	The shutter is not jammed.
WFS_CDM_SHUTTERPOS_OPEN	The shutter is jammed, but fully open.
WFS_CDM_SHUTTERPOS_PARTIALLY_OPEN	The shutter is jammed, but partially open.
WFS_CDM_SHUTTERPOS_CLOSED	The shutter is jammed, but fully closed.
WFS_CDM_SHUTTERPOS_UNKNOWN	The position of the shutter is unknown.

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

dwGuidLights [...]
Specifies the state of the guidance light indicators. The elements of this array can be accessed by using the predefined index values specified for the *dwGuidLights [ ]* field in the capabilities. Vendor specific guidance lights are defined starting from the end of the array. The maximum guidance light index is WFS_CDM_GUIDLIGHTS_MAX.

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

Value	Meaning	Type
WFS_CDM_GUIDANCE_NOT_AVAILABLE	The status is not available.	A
WFS_CDM_GUIDANCE_OFF	The light is turned off.	A
WFS_CDM_GUIDANCE_SLOW_FLASH	The light is blinking slowly.	B
WFS_CDM_GUIDANCE_MEDIUM_FLASH	The light is blinking medium frequency.	B
WFS_CDM_GUIDANCE_QUICK_FLASH	The light is blinking quickly.	B
WFS_CDM_GUIDANCE_CONTINUOUS	The light is turned on continuous (steady).	B
WFS_CDM_GUIDANCE_RED	The light is red.	C
WFS_CDM_GUIDANCE_GREEN	The light is green.	C
WFS_CDM_GUIDANCE_YELLOW	The light is yellow.	C
WFS_CDM_GUIDANCE_BLUE	The light is blue.	C
WFS_CDM_GUIDANCE_CYAN	The light is cyan.	C
WFS_CDM_GUIDANCE_MAGENTA	The light is magenta.	C
WFS_CDM_GUIDANCE_WHITE	The light is white.	C
WFS_CDM_GUIDANCE_ENTRY	The light is in the entry state.	D
WFS_CDM_GUIDANCE_EXIT	The light is in the exit state.	D

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

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

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

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

Value	Meaning
WFS_CDM_AFMNOTSUPP	No anti-fraud module is available.
WFS_CDM_AFMOK	Anti-fraud module is in a good state and no foreign device is detected.
WFS_CDM_AFMINOP	Anti-fraud module is inoperable.
WFS_CDM_AFMDEVICEDETECTED	Anti-fraud module detected the presence of a foreign device.
WFS_CDM_AFMUNKNOWN	The state of the anti-fraud module cannot be determined.

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

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

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

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

5.2 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;
     DWORD                     dwGuidLights[WFS_CDM_GUIDLIGHTS_SIZE];
     BOOL                      bPowerSaveControl;
     BOOL                      bPrepareDispense;
     BOOL                      bAntiFraudModule;
     DWORD                     dwItemInfoTypes;
     BOOL                      bBlacklist;
     LPDWORD                   lpdwSynchronizableCommands;
     BOOL                      bClassificationList;
     } WFSCDMCAPS, *LPWFSCDMCAPS;
```

wClass
Specifies the logical service class as 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 zero - 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 recycle 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. If the device does not have a retract capability this field will be WFS_CDM_RA_NOTSUPP. Otherwise this field will be set to a combination of the following flags:

Value	Meaning
WFS_CDM_RA_RETRACT	The items may be retracted to a 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 a reject cash unit.
WFS_CDM_RA_ITEMCASSETTE	The items may be retracted to the item cassettes, i.e. cassettes that can be dispensed from.

fwRetractTransportActions
Specifies the actions which may be performed on items which have been retracted to the transport. If the device does not have the capability to retract items to the transport or move items from the transport this value will be WFS_CDM_NOTSUPP. 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 moved to a retract cash unit.
WFS_CDM_REJECT	The items may be moved to a reject bin.
WFS_CDM_ITEMCASSETTE	The items may be moved to the item cassettes, i.e. cassettes that can be dispensed from.

fwRetractStackerActions
Specifies the actions which may be performed on items which have been retracted to the stacker. If the device does not have the capability to retract items to the stacker or move items from the stacker 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 moved to a retract cash unit.
WFS_CDM_REJECT	The items may be moved to a reject bin.
WFS_CDM_ITEMCASSETTE	The items may be moved to the item cassettes, i.e. cassettes that can be dispensed from.

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 field *bPresent* of the WFS_CMD_CDM_DISPENSE command can be set to FALSE.

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_ITEMSTAKEN 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 dispense 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.
WFS_CDM_TOSTACKER	The CDM can dispense items from the cash units to the intermediate stacker while there are already items on the intermediate stacker that have not been in customer access. Items remaining on the stacker from a previous dispense may first need to be rejected explicitly by the application if they are not to be presented.

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

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

In addition to supporting specific flash rates and colors, some guidance lights also have the capability to show directional movement representing “entry” and “exit”. The “entry” state gives the impression of leading a user to place media into the device .The “exit” state gives the impression of ejection from a device to a user and would be used for retrieving media from the device.

The elements of this array are specified as a combination of the following flags and indicate all of the possible flash rates (type B) colors (type C) and directions (type D) that the guidance light indicator is capable of handling. If the guidance light indicator only supports one color then no value of type C is returned. If the guidance light indicator does not support direction then no value of type D is returned. A value of WFS_CDM_GUIDANCE_NOT_AVAILABLE indicates that the device has no guidance light indicator or the device controls the light directly with no application control possible.

Value	Meaning	Type
WFS_CDM_GUIDANCE_NOT_AVAILABLE	There is no guidance light control available at this position.	A
WFS_CDM_GUIDANCE_OFF	The light can be off.	B
WFS_CDM_GUIDANCE_SLOW_FLASH	The light can blink slowly.	B
WFS_CDM_GUIDANCE_MEDIUM_FLASH	The light can blink medium frequency.	B
WFS_CDM_GUIDANCE_QUICK_FLASH	The light can blink quickly.	B
WFS_CDM_GUIDANCE_CONTINUOUS	The light can be continuous (steady).	B
WFS_CDM_GUIDANCE_RED	The light can be red.	C
WFS_CDM_GUIDANCE_GREEN	The light can be green.	C
WFS_CDM_GUIDANCE_YELLOW	The light can be yellow.	C
WFS_CDM_GUIDANCE_BLUE	The light can be blue.	C
WFS_CDM_GUIDANCE_CYAN	The light can be cyan.	C
WFS_CDM_GUIDANCE_MAGENTA	The light can be magenta.	C
WFS_CDM_GUIDANCE_WHITE	The light can be white.	C
WFS_CDM_GUIDANCE_ENTRY	The light is in the entry state.	D
WFS_CDM_GUIDANCE_EXIT	The light can be in the exit state.	D

Each array index represents an output position in the CDM. The elements are accessed using the following definitions for the index value:

Value	Meaning
WFS_CDM_GUIDANCE_POSOUTNULL	The default output position.
WFS_CDM_GUIDANCE_POSOUTLEFT	Left output position.
WFS_CDM_GUIDANCE_POSOUTRIGHT	Right output position.
WFS_CDM_GUIDANCE_POSOUTCENTER	Center output position.
WFS_CDM_GUIDANCE_POSOUTTOP	Top output position.
WFS_CDM_GUIDANCE_POSOUTBOTTOM	Bottom output position.
WFS_CDM_GUIDANCE_POSOUTFRONT	Front output position.
WFS_CDM_GUIDANCE_POSOUTREAR	Rear output position.

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

bPrepareDispense
On some hardware it can take a significant amount of time for the dispenser to get ready to dispense media. On this type of hardware the WFS_CMD_CDM_PREPARE_DISPENSE command can be used to improve transaction performance. This flag indicates if the hardware requires the application to use the WFS_CMD_CDM_PREPARE_DISPENSE command to maximize transaction performance. If this flag is TRUE then the WFS_CMD_CDM_PREPARE_DISPENSE command is supported and can be used to improve transaction performance. If this flag is FALSE then the WFS_CMD_CDM_PREPARE_DISPENSE command is not supported.

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

dwItemInfoTypes
Specifies the types of information that can be retrieved through the WFS_INF_CDM_GET_ITEM_INFO command as a combination of the following flags:

Value	Meaning
WFS_CDM_ITEM_SERIALNUMBER	Serial Number of the item.
WFS_CDM_ITEM_SIGNATURE	Signature of the item.
WFS_CDM_ITEM_IMAGEFILE	Image file of the item.

bBlacklist
Specifies whether the device has the capability to maintain a blacklist of serial numbers as well as supporting the associated operations. This can either be TRUE if the device has the capability or FALSE if it does not.

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

bClassificationList
Specifies whether the device has the capability to maintain a classification list of serial numbers as well as supporting the associated operations. This can either be TRUE if the device has the capability or FALSE if it does not.

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

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

5.3 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 or if it can store retracted and rejected items as separate logical units and report meaningful count and replenishment information for each of them. 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:

lpszCashUnitName
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

Item counts are typically based on 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. If a cash unit is shared between the CDM and CIM device class, then CDM operations will result in count changes in the CIM cash unit structure and vice versa. All counts are reported consistently on both interfaces at all times.

On cash units that dispense items, if *ulCount* (on logical and physical cash units) reaches zero it will not decrement further but will remain at zero. When *ulCount* reaches zero no further dispense or denominate operations will be possible using that cash unit, unless the Service Provider provides a configuration option to continue using cash units when *ulCount* reaches zero. The default setting for any such configuration parameter must be to stop using the cash unit when this value reaches zero. If the Service Provider is configured such that the cash unit can still be used when *ulCount* reaches zero then WFS_CDM_STATCUEMPTY should not be generated when *ulCount* reaches zero, rather it should be generated when all physical cash units associated with the logical cash unit are physically empty. On recyclers, the Service Provider should not be configured to keep using the cash unit when *ulCount* is zero if the value in *ulCount* is used by any part of the application, as it may not be accurate. However, if the Service Provider is configured to keep using the cash unit when *ulCount* reaches zero, then the number of notes in the cash unit can be determined relative to *ulInitialCount* using *ulDispensedCount*, *ulRetractedCount* and the CIM *ulCashInCount*, e.g. Number of Notes = *ulInitialCount – ulDispensedCount* + *ulRetractedCount* + CIM::*ulCashInCount*.

On cash units that dispense items, *ulCount* on logical cash units is decremented when items are in customer access or successfully rejected. Therefore, items which are dispensed from the cash unit then removed from the device for other reasons (such as manual jam clearance) may mean that

ulCount on logical cash units no longer reflects the number of items in the cash unit. The count of items in the cash unit should therefore be determined from other counts, e.g., *ulCount* on physical cash units or Number of Notes = *ulInitialCount* – *ulDispensedCount* + *ulRetractedCount* + CIM::*ulCashInCount* (if applicable).

Exchanges

If a physical cash unit is inserted (including removal followed by a reinsertion) when the device is not in the exchange state the *usStatus* of the physical cash unit will be set to WFS_CDM_STATCUMANIP 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.

On recycling and retract units the counts and status are consistently reported on both the CDM and CIM interfaces. When a value is changed through an exchange on one interface it is also changed on the other.

Recyclers

The CDM interface does not report cash-in only cash units but does report cash units which are shared with the CIM, i.e. recycling cash units (WFS_CDM_TYPERECYCLING) and reject/retract cash units (WFS_CDM_TYPEREJECTCASSETTE / WFS_CDM_TYPERETRACTCASSETTE). The CIM interface reports all cash units of all types, including those that can only be used by commands on the CDM interface.

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 zero.

usCount
Specifies the number of cash unit structures returned.

lppList
Pointer to an array of pointers to WFSCDMCASHUNIT 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;
     ULONG                     ulDispensedCount;
     ULONG                     ulPresentedCount;
     ULONG                     ulRetractedCount;
     } 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. This type will also indicate a combined reject/retract 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.

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, travelers checks and bank checks 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. This value is persistent.

cUnitID
The Cash Unit Identifier.

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. This value is persistent.

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 an array of three ASCII 0x20h characters, then this field will contain zero. 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. This value is persistent.

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

ulCount
The meaning of this count depends on the type of cash unit. This value is persistent.

For all cash units except retract cash units (*usType* is not WFS_CDM_TYPERETRACTCASSETTE) this value specifies the number of items inside all the physical cash units associated with this cash unit.

For all dispensing cash units (*usType* is WFS_CDM_TYPEBILLCASSETTE, WFS_CDM_TYPECOINCYLINDER, WFS_CDM_TYPECOINDISPENSER, WFS_CDM_TYPECOUPON, WFS_CDM_TYPEDOCUMENT or WFS_CDM_TYPERECYCLING), this value includes any items from the physical cash units not yet presented to the customer. This count is only decremented when the items are either known to be in customer access or successfully rejected.

If the cash unit is usable from the CIM interface (*usType* is WFS_CDM_TYPERECYCLING, WFS_CDM_TYPERETRACTCASSETTE or WFS_CDM_TYPEREJECTCASSETTE) then this value will be incremented as a result of a cash-in operation.

Note that for a reject cash unit (*usType* is WFS_CDM_TYPEREJECTCASSETTE), 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 (*usType* is WFS_CDM_TYPERETRACTCASSETTE) this value specifies the number of retract operations (CDM commands, CIM commands and error recoveries) which result in items entering the cash unit.

ulRejectCount
The number of items dispensed from this cash unit which have been rejected, are in a cash unit other than this cash unit, and which have not been accessible to a customer. This value may be unreliable, since a typical reason for rejecting items is a suspected pick failure. Other reasons for rejecting items may include incorrect note denominations, classifications not valid for dispensing, or where the transaction has been cancelled and WFS_CMD_CDM_REJECT has been called. For reject and retract cash units (*usType* is WFS_CDM_TYPEREJECTCASSETTE or WFS_CDM_TYPERETRACTCASSETTE) this field does not apply and will be reported as zero. 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 (WFS_CDM_STATCULOW) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if *bHardwareSensor* is TRUE. This value is persistent.

ulMaximum
This field is only applicable to retract and reject cash units. When *ulCount* reaches this value the threshold event WFS_USRE_CDM_CASHUNITTHRESHOLD (WFS_CDM_STATCUHIGH) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if *bHardwareSensor* is TRUE. This value is persistent.

bAppLock
If this value is TRUE items cannot be dispensed from or deposited into the cash unit. If this value is TRUE and the application attempts to use the cash unit a WFS_EXEE_CDM_CASHUNITERROR event will be generated and a WFS_ERR_CDM_CASHUNITERROR code will be returned. This value is persistent.

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. This value only applies to cash units where *usType* is WFS_CDM_TYPEREJECTCASSETTE or WFS_CDM_TYPERETRACT-CASSETTE.

WFS_CDM_STATCUHIGH	The cash unit is almost full (i.e. reached or exceeded the threshold defined by *ulMaximum*). This value only applies to cash units where *usType* is WFS_CDM_TYPEREJECTCASSETTE or WFS_CDM_TYPERETRACT-CASSETTE.
WFS_CDM_STATCULOW	The cash unit is almost empty (i.e. reached or below the threshold defined by *ulMinimum*). This value does not apply to cash units where *usType* is WFS_CDM_TYPEREJECTCASSETTE or WFS_CDM_TYPERETRACT-CASSETTE.
WFS_CDM_STATCUEMPTY	The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations. Note that this status is available to all cash unit types, which is a change from previous versions of this specification, therefore service providers may wish to make this a configurable option.
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 inserted (including removal followed by a reinsertion) when the device was not in the exchange state. This cash unit cannot be dispensed from.

ulDispensedCount
The number of items dispensed from all the physical cash units associated with this cash unit. This count is incremented when the items are removed from any of the associated physical cash units. This count includes any items that were rejected during the dispense operation and are no longer in this cash unit. This field is always zero for cash units with a *usType* of WFS_CDM_TYPEREJECTCASSETTE or WFS_CDM_TYPERETRACTCASSETTE. This value is persistent.

ulPresentedCount
The number of items from all the physical cash units associated with this cash unit that have been presented to the customer. This count is incremented when the items are presented to the customer. If it is unknown if a customer has been presented with the items, then this count is not updated. This field is always zero for cash units with a *usType* of WFS_CDM_TYPEREJECTCASSETTE or WFS_CDM_TYPERETRACTCASSETTE. This value is persistent.

ulRetractedCount
The number of items that have been accessible to a customer and retracted into all the physical cash units associated with this cash unit. This value is persistent.

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 WFSCDMPHCU structures:

```
typedef struct _wfs_cdm_physicalcu
     {
     LPSTR                     lpPhysicalPositionName;
     CHAR                      cUnitID[5];
     ULONG                     ulInitialCount;
     ULONG                     ulCount;
     ULONG                     ulRejectCount;
     ULONG                     ulMaximum;
     USHORT                    usPStatus;
     BOOL                      bHardwareSensor;
     ULONG                     ulDispensedCount;
     ULONG                     ulPresentedCount;
     ULONG                     ulRetractedCount;
     } 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. This value is persistent.

ulCount
As defined by the logical *ulCount* description but applies to a single physical cash unit, but with the following exceptions:

This count does not include items dispensed but not yet presented.

On cash units belonging to logical cash units with *usType* set to WFS_CDM_TYPERETRACTCASSETTE the physical count represents the number of items, unless the device cannot count items during a retract, in which case this count will be zero.

This value is persistent.

ulRejectCount
As defined by the logical *ulRejectCount* description but applies to a single physical cash unit. 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. This value is persistent.

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. This value only applies to cash units where *usType* is WFS_CDM_TYPEREJECT-CASSETTE or WFS_CDM_TYPERETRACT-CASSETTE.
WFS_CDM_STATCUHIGH	The cash unit is almost full (reached or exceeded threshold defined by *ulMaximum*). This value only applies to cash units where *usType* is WFS_CDM_TYPEREJECT-CASSETTE or WFS_CDM_TYPERETRACT-CASSETTE.

WFS_CDM_STATCULOW	The cash unit is almost empty. This value does not apply to cash units where *usType* is WFS_CDM_TYPE-REJECTCASSETTE or WFS_CDM_TYPERETRACT-CASSETTE.
WFS_CDM_STATCUEMPTY	The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations. Note that this status is available to all cash unit types, which is a change from previous versions of this specification, therefore service providers may wish to make this a configurable option.
WFS_CDM_STATCUINOP	The cash unit is inoperative.
WFS_CDM_STATCUMISSING	The cash unit is missing. The cash unit has been removed and is physically not present in the machine.
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 inserted (including removal followed by a reinsertion) 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.

ulDispensedCount
As defined by the logical *ulDispensedCount* description but applies to a single physical cash unit. This value is zero if the h/w does not support physical counts. This value is persistent.

ulPresentedCount
As defined by the logical *ulPresentedCount* description but applies to a single physical cash unit. This value is zero if the h/w does not support physical counts. This value is persistent.

ulRetractedCount
As defined by the logical *ulRetractedCount* description but applies to a single physical cash unit. This value is zero if the h/w does not support physical counts. This value is persistent.

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

Comments None.

5.4 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 field can be an array of three ASCII 0x20 characters. In this case information on all currencies will be returned.

Output Param LPWFSCDMTELLERDETAILS *lppTellerDetails;

Pointer to a NULL-terminated array of pointers to WFSCDMTELLERDETAILS 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 WFSCDMTELLERTOTALS 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 section 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 section WFS_INF_CDM_CURRENCY_EXP).

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

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

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

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

5.5 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 WFSCDMCURRENCYEXP structures:

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

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

sExponent
Currency exponent in ISO 4217 format [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. European Euro, Japanese Yen). 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 ten to the power of the currency exponent. Thus an amount parameter relates to the actual cash amount as follows:

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

Example #1 - Euro
Currency identifier is 'EUR'
Currency unit is 1 Euro (= 100 Cent)

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

Example #2 - Japan
Currency identifier is 'JPY'
Currency unit is 1 Japanese Yen

A Service Provider is required to dispense a minimum amount of 1000 Yen. The currency exponent (*sExponent*) is set to +3 (plus three), so the minimum dispense unit is 1000 Yen; all amounts at the XFS interface are in multiples of 1000 Yen. Thus an amount parameter of 15 is 15000 Yen.

5.6 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 WFSCDMMIXTYPE 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. 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 - 8FFF. Application defined mix algorithms start at hexadecimal 9000. All numbers below 8000 hexadecimal are reserved. If *usMixType* is WFS_CDM_MIXTABLE, this value will be zero. 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 logical cash units are used.

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.

5.7 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
Pointer 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 house mix 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 section 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 section WFS_INF_CDM_CURRENCY_EXP).

lpusMixture
Pointer to a mix row, an array of length *usCols* of USHORTs; 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.

Comments None.

5.8 WFS_INF_CDM_PRESENT_STATUS

Description This command is used to obtain the status of the most recent attempt to dispense and/or present items to the customer from a specified output position. The items may have been dispensed and/or presented as a result of the WFS_CMD_CDM_PRESENT or WFS_CMD_CDM_DISPENSE command. This status is not updated as a result of any other command that can dispense/present items.

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

The denominations reported by this command may not accurately reflect the operation if the cash units have been re-configured (e.g. if the values associated with a cash unit are changed, or new cash units are configured).

Input Param LPWORD lpfwPosition;

lpfwPosition
Pointer to the required output position 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.

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 amount dispensed from the specified output position and the number of items dispensed from each cash unit. For a description of the WFSCDMDENOMINATION structure see the definition of the command WFS_CMD_CDM_DENOMINATE. Where the capability *fwMoveItems* reports WFS_CDM_TOSTACKER this value is cumulative across a series of WFS_CMD_CDM_DISPENSE calls that add additional items to the stacker.

Where mixed currencies were dispensed the *ulAmount* field in the returned denomination structure will be zero and the *cCurrencyID* field will be set to three ASCII 0x20 characters.

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 has not had access to the items.
WFS_CDM_UNKNOWN	It is not known if the customer had access to the items.

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

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 specified output position is not supported.

Comments None.

5.9 WFS_INF_CDM_GET_ITEM_INFO

Description This command is used to get information about a single detected item. This information is available from the point where the first WFS_EXEE_CDM_INFO_AVAILABLE event is generated until one of the following CDM commands is executed:

WFS_CMD_CDM_DISPENSE, WFS_CMD_CDM_COUNT, WFS_CMD_CDM_PRESENT, WFS_CMD_CDM_RETRACT, WFS_CMD_CDM_REJECT, WFS_CMD_CDM_OPEN_SHUTTER, WFS_CMD_CDM_CLOSE_SHUTTER, WFS_CMD_CDM_RESET, WFS_CMD_CDM_START_EXCHANGE, WFS_CMD_CDM_END_EXCHANGE, WFS_CMD_CDM_CALIBRATE_CASH_UNIT, WFS_CMD_CDM_TEST_CASH_UNITS.

Additionally for a recycler, the following CIM commands will also invalidate the information:

WFS_CMD_CIM_CASH_IN_START, WFS_CMD_CIM_CASH_IN, WFS_CMD_CIM_CASH_IN_ROLLBACK, WFS_CMD_CIM_CASH_IN_END, WFS_CMD_CIM_RETRACT, WFS_CMD_CIM_RESET, WFS_CMD_CIM_START_EXCHANGE, WFS_CMD_CIM_END_EXCHANGE, WFS_CMD_CIM_CREATE_P6_SIGNATURE, WFS_CMD_CIM_REPLENISH, WFS_CMD_CIM_CASH_UNIT_COUNT.

This command is used to retrieve the required information on an individual item basis. Applications should loop retrieving the information for each index and for each level reported with the WFS_EXEE_CDM_INFO_AVAILABLE event.

Input Param LPWFSCDMGETITEMINFO lpGetItemInfo;

```
typedef struct _wfs_cdm_get_item_info
     {
     USHORT                    usLevel;
     USHORT                    usIndex;
     DWORD                     dwItemInfoType;
     } WFSCDMGETITEMINFO, *LPWFSCDMGETITEMINFO;
```

usLevel
Defines the note level. Possible values are:

Value	Meaning
WFS_CDM_LEVEL_1	Information for level 1 notes. Only an image file can be retrieved for level 1 notes.
WFS_CDM_LEVEL_2	Information for level 2 notes. On systems that do not classify notes as level 2 this value cannot be used and WFS_ERR_INVALID_DATA will be returned.
WFS_CDM_LEVEL_3	Information for level 3 notes. On systems that do not classify notes as level 3 this value cannot be used and WFS_ERR_INVALID_DATA will be returned.
WFS_CDM_LEVEL_4	Information for level 4 notes.

usIndex
Specifies the index for the item information required (zero to *usNumOfItems*-1 as reported in the WFS_EXEE_CDM_INFO_AVAILABLE event).

dwItemInfoType
Specifies the type of information required. This can be a combination of the following flags:

Value	Meaning
WFS_CDM_ITEM_SERIALNUMBER	Serial number of the item.
WFS_CDM_ITEM_SIGNATURE	Signature of the item.
WFS_CDM_ITEM_IMAGEFILE	Image file of the item.

Output Param LPWFSCDMITEMINFO lpItemInfo;

The data returned by this command relates to a single item (*usIndex*).

```
typedef struct _wfs_cdm_item_info
     {
     CHAR                      cCurrencyID[3];
     ULONG                     ulValue;
     USHORT                    usRelease;
     LPWSTR                    lpszSerialNumber;
     LPWFSCDMSIGNATURE         lpSignature;
     LPSTR                     lpszImageFileName;
     } WFSCDMITEMINFO, *LPWFSCDMITEMINFO;
```

cCurrencyID
Currency ID in ISO 4217 format [Ref. 2]. This value will be an array of three ASCII 0x20h characters for level 1 items.

ulValue
The value of a single item expressed in minimum dispense units. This value will be zero for level 1 items.

usRelease
The release of the banknote type. The higher this number is, the newer the release. Zero means that there is only one release of that banknote type. This value has not been standardized and therefore a release number of the same banknote will not necessarily have the same value in different systems. This value will be zero for level 1 items.

lpszSerialNumber
This field contains the serial number of the item as a Unicode string. A '?' character (0x003F) is used to represent any serial number character that cannot be recognized. If no serial number is available or has not been requested then *lpszSerialNumber* is NULL.

lpSignature
This field contains the signature for the item. If no signature is available or has not been requested then this field is NULL.

```
typedef struct _wfs_cdm_signature
     {
     ULONG                     ulLength;
     LPVOID                    lpData;
     } WFSCDMSIGNATURE, *LPWFSCDMSIGNATURE;
```

ulLength
Length of the signature in bytes.

lpData
Pointer to the returned signature data.

lpszImageFileName
Name of the file where the image containing the item's serial number is stored, e.g. "C:\Temp\cash123456.jpg". If the Service Provider does not support this function or the image file has not been requested then *lpszImageFileName* is NULL.
The application is responsible for the use and management of this file. For example, the application can transfer the image files to a directory which is managed by the application.

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

Comments The application has to call this command multiple times in a loop where there is multiple information to retrieve. In addition, since the item information is not cumulative and can be replaced by any command that can move notes, it is recommended that applications that are interested in the available information should query for it following the WFS_EXEE_CDM_INFO_AVAILABLE event but before any other command is executed.

5.10 WFS_INF_CDM_GET_BLACKLIST

Description This command is used to retrieve the entire blacklist information pre-set inside the device or set via the WFS_CMD_CDM_SET_BLACKLIST or WFS_CMD_CDM_SET_CLASSIFICATION_LIST command, or WFS_CMD_CIM_SET_BLACKLIST or WFS_CMD_CIM_SET_CLASSIFICATION_LIST in the case of a recycler.

Input Param None.

Output Param LPWFSCDMBLACKLIST lpBlacklist;

```
typedef struct _wfs_cdm_blacklist
     {
     LPWSTR                    lpszVersion;
     USHORT                    usCount;
     LPWFSCDMBLACKLISTELEMENT  *lppBlacklistElements;
     } WFSCDMBLACKLIST, *LPWFSCDMBLACKLIST;
```

lpszVersion
This is an application defined Unicode string that sets the version identifier of the blacklist. This can be set to NULL if it has no version identifier.

usCount
Number of pointers to WFSCDMBLACKLISTELEMENT structures returned in *lppBlacklistElements*.

lppBlacklistElements
Pointer to an array of pointers to WFSCDMBLACKLISTELEMENT structures.

```
typedef struct _wfs_cdm_blacklist_element
     {
     LPWSTR                    lpszSerialNumber;
     CHAR                      cCurrencyID[3];
     ULONG                     ulValue;
     } WFSCDMBLACKLISTELEMENT, *LPWFSCDMBLACKLISTELEMENT;
```

lpszSerialNumber
This Unicode string defines the serial number or a mask of serial numbers of one blacklist item with the defined currency and value. For a definition of the mask see Section 4.

cCurrencyID
The three character ISO format currency identifier [Ref. 2] of the blacklist element.

ulValue
The value of a blacklist element. This field can be zero to represent all values.

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

Comments None.

5.11 WFS_INF_CDM_GET_ALL_ITEMS_INFO

Description This command can be used to retrieve all item information available for all levels at once by specifying WFS_CDM_LEVEL_ALL in the *usLevel* parameter. Alternatively this command can be used to retrieve all information for a particular level of banknote. This information is available from the point where the first WFS_EXEE_CDM_INFO_AVAILABLE event is generated until one of the following CDM commands is executed:

WFS_CMD_CDM_DISPENSE, WFS_CMD_CDM_COUNT, WFS_CMD_CDM_PRESENT, WFS_CMD_CDM_RETRACT, WFS_CMD_CDM_REJECT, WFS_CMD_CDM_OPEN_SHUTTER, WFS_CMD_CDM_CLOSE_SHUTTER, WFS_CMD_CDM_RESET, WFS_CMD_CDM_START_EXCHANGE, WFS_CMD_CDM_END_EXCHANGE, WFS_CMD_CDM_CALIBRATE_CASH_UNIT, WFS_CMD_CDM_TEST_CASH_UNITS.

Additionally for a recycler, the following CIM commands will also invalidate the information:

WFS_CMD_CIM_CASH_IN_START, WFS_CMD_CIM_CASH_IN, WFS_CMD_CIM_CASH_IN_ROLLBACK, WFS_CMD_CIM_CASH_IN_END, WFS_CMD_CIM_RETRACT, WFS_CMD_CIM_RESET, WFS_CMD_CIM_START_EXCHANGE, WFS_CMD_CIM_END_EXCHANGE, WFS_CMD_CIM_CREATE_P6_SIGNATURE, WFS_CMD_CIM_REPLENISH, WFS_CMD_CIM_CASH_UNIT_COUNT.

The WFS_EXEE_CDM_INPUT_P6 event is only generated when level 2 and/or level 3 notes are detected.

Input Param LPWFSCDMGETALLITEMSINFO lpGetAllItemsInfo;

```
typedef struct _wfs_cdm_get_all_items_info
     {
     USHORT                    usLevel;
     } WFSCDMGETALLITEMSINFO, *LPWFSCDMGETALLITEMSINFO;
```

usLevel
Defines the note level. Possible values are:

Value	Meaning
WFS_CDM_LEVEL_1	Information for level 1 notes. Only an image file can be retrieved for level 1 notes.
WFS_CDM_LEVEL_2	Information for level 2 notes is to be returned with the *lpAllItemsInfo* output parameter. On systems that do not classify notes as level 2 this value cannot be used and WFS_ERR_INVALID_DATA will be returned.
WFS_CDM_LEVEL_3	Information for level 3 notes is to be returned with the *lpAllItemsInfo* output parameter. On systems that do not classify notes as level 3 this value cannot be used and WFS_ERR_INVALID_DATA will be returned.
WFS_CDM_LEVEL_4	Information for level 4 notes is to be returned with the *lpAllItemsInfo* output parameter.
WFS_CDM_LEVEL_ALL	Information for all levels all items is to be returned with the *lpAllItemsInfo* output parameter.

Output Param LPWFSCDMALLITEMSINFO lpAllItemsInfo;

```
typedef struct _wfs_cdm_all_items_info
     {
     USHORT                    usCount;
     LPWFSCDMITEMINFOALL       *lppItemsList;
     } WFSCDMALLITEMSINFO, *LPWFSCDMALLITEMSINFO;
```

usCount
Number of WFSCDMITEMINFOALL structures returned in *lppItemsList*.

lppItemsList
Pointer to an array of pointers to WFSCDMITEMINFOALL structures:

```
typedef struct _wfs_cdm_item_info_all
     {
     USHORT                    usLevel;
     CHAR                      cCurrencyID[3];
     ULONG                     ulValue;
     USHORT                    usRelease;
     LPWSTR                    lpszSerialNumber;
     LPSTR                     lpszImageFileName;
     WORD                      wOnBlacklist;
     WORD                      wItemLocation;
     USHORT                    usNumber;
     WORD                      wOnClassificationList;
     WORD                      wItemDeviceLocation;
     } WFSCDMITEMINFOALL, *LPWFSCDMITEMINFOALL;
```

usLevel
Defines the note level. Possible values are:

Value	Meaning
WFS_CDM_LEVEL_1	A level 1 banknote.
WFS_CDM_LEVEL_2	A level 2 banknote.
WFS_CDM_LEVEL_3	A level 3 banknote.
WFS_CDM_LEVEL_4	A level 4 banknote.

cCurrencyID
Currency ID in ISO 4217 format [Ref. 2]. This value will be an array of three ASCII 0x20h characters for level 1 items.

ulValue
The value of a single item expressed in minimum dispense units. This value will be zero for level 1 items.

usRelease
The release of the banknote type. The higher this number is, the newer the release. Zero means that there is only one release of that banknote type. This value has not been standardized and therefore a release number of the same banknote will not necessarily have the same value in different systems. This value will be zero for level 1 items.

lpszSerialNumber
This field contains the serial number of the item as a Unicode string. A '?' character (0x003F) is used to represent any serial number character that cannot be recognized. If no serial number is available then *lpszSerialNumber* is NULL.

lpszImageFileName
Full file path to an image file containing the serial number(s). If no image is available then this field is NULL. The application is responsible for the use and management of this file. For example, the application can transfer the image files to a directory which is managed by the application.

wOnBlacklist
Specifies if the serial number reported in the *lpszSerialNumber* field is on the blacklist. If the blacklist reporting capability is not supported this field will be zero. Otherwise, possible values are:

Value	Meaning
WFS_CDM_ONBLACKLIST	The serial number of the items is on the blacklist.
WFS_CDM_NOTONBLACKLIST	The serial number of the items is not on the blacklist.
WFS_CDM_BLACKLISTUNKNOWN	It is unknown if the serial number of the item is on the blacklist.

wItemLocation
Specifies the location of the item as one of the following values:

Value	Meaning
WFS_CDM_LOCATION_DEVICE	The item is inside the device in some position other than a cash unit.
WFS_CDM_LOCATION_CASHUNIT	The item is in a cash unit. The logical cash unit number is defined by *usNumber*.
WFS_CDM_LOCATION_CUSTOMER	The item has been dispensed to the customer.
WFS_CDM_LOCATION_UNKNOWN	The item location is unknown.

usNumber
If *wItemLocation* is WFS_CDM_LOCATION_CASHUNIT this parameter specifies the logical number of the cash unit which received the item. If *wItemLocation* is not WFS_CDM_LOCATION_CASHUNIT then *usNumber* will be zero.

wOnClassificationList
Specifies if the serial number reported in the *lpszSerialNumber* field is on the classification list. If the classification list reporting capability is not supported this field will be zero. Otherwise, possible values are:

Value	Meaning
WFS_CDM_CLASSIFICATIONLIST_ON	The serial number of the items is on the classification list.
WFS_CDM_CLASSIFICATIONLIST_NOTON	The serial number of the items is not on the classification list.
WFS_CDM_CLASSIFICATIONLIST_UNKNOWN	It is unknown if the serial number of the item is on the classification list.

wItemDeviceLocation
If *wItemLocation* is WFS_CDM_LOCATION_DEVICE this parameter specifies where the item is in the device. If *wItemLocation* is not WFS_CDM_LOCATION_DEVICE then *wItemDeviceLocation* will be zero:

Value	Meaning
WFS_CDM_DEVLOC_STACKER	The item is in the intermediate stacker.
WFS_CDM_DEVLOC_OUTPUT	The item is at the output position. The items have not been in customer access.
WFS_CDM_DEVLOC_TRANSPORT	The item is at another location in the device.
WFS_CDM_DEVLOC_UNKNOWN	The item is in the device but its location is unknown.

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

Comments In addition, since the item information is not cumulative and can be replaced by any command that can move notes, it is recommended that applications that are interested in the available information should query for it following the WFS_EXEE_CDM_INFO_AVAILABLE event but before any other command is executed.

5.12 WFS_INF_CDM_GET_CLASSIFICATION_LIST

Description This command is used to retrieve the entire note classification information pre-set inside the device or set via the WFS_CMD_CDM_SET_CLASSIFICATION_LIST or WFS_CMD_CDM_SET_BLACKLIST command, or WFS_CMD_CIM_SET_CLASSIFICATION_LIST or WFS_CMD_CIM_SET_BLACKLIST in the case of a recycler.

This extends the functionality provided by the blacklist commands and allows additional flexibility, for example to specify that notes can be taken out of circulation by specifying them as unfit. Any items not returned in this list will be handled according to normal classification rules.

Input Param None.

Output Param LPWFSCDMCLASSIFICATIONLIST lpClassificationList;

```
typedef struct _wfs_cdm_classification_list
     {
     LPWSTR                          lpszVersion;
     USHORT                          usCount;
     LPWFSCDMCLASSIFICATIONELEMENT   *lppClassificationElements;
     } WFSCDMCLASSIFICATIONLIST, *LPWFSCDMCLASSIFICATIONLIST;
```

lpszVersion
This is an application defined Unicode string that sets the version identifier of the classification list. This can be set to NULL if it has no version identifier.

usCount
Number of pointers to WFSCDMCLASSIFICATIONELEMENT structures returned in *lppClassificationElements*.

lppClassificationElements
Pointer to an array of pointers to WFSCDMCLASSIFICATIONELEMENT structures.

```
typedef struct _wfs_cdm_classification_element
     {
     LPWSTR                    lpszSerialNumber;
     CHAR                      cCurrencyID[3];
     ULONG                     ulValue;
     USHORT                    usLevel;
     BOOL                      bUnfit;
     } WFSCDMCLASSIFICATIONELEMENT, *LPWFSCDMCLASSIFICATIONELEMENT;
```

lpszSerialNumber
This Unicode string defines the serial number or a mask of serial numbers of one element with the defined currency and value. For a definition of the mask see Section 4.

cCurrencyID
The three character ISO format currency identifier [Ref. 2] of the element.

ulValue
The value of the element. This field can be zero to represent all values.

usLevel
Specifies the note level. Possible values are:

Value	Meaning
WFS_CDM_LEVEL_1	The element specifies notes to be treated as level 1 notes.
WFS_CDM_LEVEL_2	The element specifies notes to be treated as level 2 notes.
WFS_CDM_LEVEL_3	The element specifies notes to be treated as level 3 notes.
WFS_CDM_LEVEL_4	The element specifies notes to be treated as level 4 notes.

bUnfit
Specifies whether the item is to be treated as unfit for dispensing. Applies only where *usLevel* is WFS_CDM_LEVEL_4.

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

Comments None.

6. Execute Commands

6.1 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 zero 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 zero. 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, currency 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 field 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 [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 zero.

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_END_EXCHANGE command. If this value is not required because a mix algorithm is used then the *usCount* can be set to zero.

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 by 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 its 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 zero and the *cCurrencyID* 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 *cCurrencyID* field of the input parameter.
WFS_ERR_CDM_INVALIDTELLERID	Invalid teller ID. This error will never be generated by a Self-Service CDM.
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 *ulCashBox* and the items specified by *lpulValues* does not match the non-zero amount specified. This error code is not used when the amount specified is zero.
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. This error code is also returned if the *usMixNumber* is specified as WFS_CDM_INDIVIDUAL, but a cash unit is specified in the *lpulValues* list which is not a dispensing cash unit, e.g., a retract/reject cash unit.
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 section 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.

6.2 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 zero 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 zero, 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 cash box 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 field *fwPosition* 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* field is ignored. If the CDM does not have an intermediate stacker then *bPresent* is ignored.

If *bPresent* is set to TRUE and a shutter exists, then it will be implicitly controlled during the present operation, even if the *bShutterControl* capability is set to FALSE. The shutter will be closed when the user removes the items or the items are retracted.

Note that a level 4 note can be dispensed, but is not necessarily presented to the customer. e.g. a note can be skewed, or can be unfit for dispensing.

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 field 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 zero and the *cCurrencyID* 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 *cCurrencyID* field of the input parameter.
WFS_ERR_CDM_INVALIDTELLERID	Invalid teller ID. This error will never be generated by a Self-Service CDM.
WFS_ERR_CDM_CASHUNITERROR	There is a problem with a cash unit. A 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. This error code is also returned if the *usMixNumber* is specified as WFS_CDM_INDIVIDUAL, but a cash unit is specified in the *lpulValues* list which is not a dispensing cash unit, e.g., a retract/reject cash unit.

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. This device requires the safe door to be closed in order to perform this operation.
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.
WFS_ERR_CDM_SHUTTEROPEN	The Service Provider cannot dispense items with an open output shutter.

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

Value	Meaning
WFS_ERR_CDM_SHUTTERNOTOPEN	The shutter is not open or did not open when it should have. 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 requested denomination is on the intermediate stacker or in customer access. The error code will be WFS_ERR_CDM_NOTDISPENSABLE.
WFS_EXEE_CDM_NOTEERROR	An item detection error has occurred.
WFS_EXEE_CDM_INPUT_P6	Level 2 and/or level 3 notes have been detected.
WFS_EXEE_CDM_INFO_AVAILABLE	Information is available for items being processed by this operation.
WFS_SRVE_CDM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.3 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;

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

Output Param LPWFSCDMCOUNT lpCount;

```
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
Specifies 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 (reached or exceeded the threshold defined by WFSCDMCASHUNIT.*ulMaximum*).
WFS_CDM_STATCULOW	The cash unit is almost empty.
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 inserted (including removal followed by a reinsertion) 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. This device requires the safe door to be closed in order to perform this operation.
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_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.
WFS_EXEE_CDM_NOTEERROR	An item detection error has occurred.
WFS_EXEE_CDM_INPUT_P6	Level 2 and/or level 3 notes have been detected.

WFS_EXEE_CDM_INFO_AVAILABLE	Information is available for items being processed by this operation.
WFS_SRVE_CDM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.4 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, even if the *bShutterControl* capability is set to FALSE. The shutter will be closed when the user removes the items or the items are retracted. If *lpfwPosition* points to WFS_CDM_POSNULL the position set in the WFS_CMD_CDM_DISPENSE command which caused these items to be dispensed will be used.

In the case where the shutter is unlocked but deliberately held shut, if the items could have been in customer access then a WFS_ERR_CDM_PRERRORITEMS error code will be returned.

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

Input Param LPWORD lpfwPosition;

lpfwPosition
Pointer to the output position where 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 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.
WFS_ERR_CDM_UNSUPPOSITION	The position specified 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 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.
WFS_EXEE_CDM_INPUT_P6	Level 2 and/or level 3 notes have been detected.

WFS_EXEE_CDM_INFO_AVAILABLE	Information is available for items being processed by this operation.
WFS_SRVE_CDM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.5 WFS_CMD_CDM_REJECT

Description This command will move items from the intermediate stacker and transport them to a reject cash unit (i.e. a cash unit with *usType* WFS_CDM_TYPEREJECTCASSETTE). The WFSCDMCASHUNIT.*ulCount* field of the reject cash unit is incremented by the number of items that were thought to be present at the time of the reject or the number counted by the device during the reject. Note that the reject bin count is unreliable.

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	A 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 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.
WFS_EXEE_CDM_INPUT_P6	Level 2 and/or level 3 notes have been detected.
WFS_EXEE_CDM_INFO_AVAILABLE	Information is available for items being processed by this operation.

Comments None.

6.6 WFS_CMD_CDM_RETRACT

Description This command will retract items which may have been in customer access from an output position or from internal areas within the CDM. Retracted items will be moved to either a retract cash unit, a reject cash unit, item cash units, the transport or the intermediate stacker. After the items are retracted the shutter is closed automatically, even if the *bShutterControl* capability is set to FALSE.

If items are moved to a retract cash unit (i.e. a cash unit with *usType* WFS_CDM_TYPERETRACTCASSETTE), then the WFSCDMCASHUNIT.*ulCount* field of the retract cash unit must be incremented by 1 to specify the number of retracts. If items are moved to any other cash unit (e.g. a cash unit with *usType* WFS_CDM_TYPEREJECTCASSETTE) then the WFSCDMCASHUNIT.*ulCount* field of the cash unit must be incremented by the number of items that were thought to be present at the time the WFS_CMD_CDM_RETRACT command was issued or the number counted by the device during the retract. Note that reject bin counts are unreliable.

Input Param LPWFSCDMRETRACT lpRetract;

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

fwOutputPosition
Specifies the output position from which to retract the items. The value is set to one of the following values:

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.
WFS_CDM_RA_ITEMCASSETTE	Retract the items to the item cassettes, i.e. cassettes that can be dispensed from.

usIndex
If *usRetractArea* is set to WFS_CDM_RA_RETRACT this field defines the position inside the retract cash units into which the cash is to be retracted. *usIndex* starts with a value of one (1) for the first retract position and increments by one for each subsequent position. If there are 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 WFSCDMCASHUNIT.*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 LPWFSCDMITEMNUMBERLIST lpItemNumberList;

Pointer to a WFSCDMITEMNUMBERLIST structure. This parameter will provide details about the items moved with this command or this parameter will be NULL if the device is not capable of identifying the moved items.

```
typedef struct _wfs_cdm_item_number_list
     {
     USHORT                          usNumOfItemNumbers;
     LPWFSCDMITEMNUMBER              *lppItemNumber;
     } WFSCDMITEMNUMBERLIST, *LPWFSCDMITEMNUMBERLIST;
```

usNumOfItemNumbers
Number of item types moved during this command, i.e. the number of *lppItemNumber* list elements.

lppItemNumber
List of item types moved to the *usRetractArea* during this command. A pointer to an array of pointers to WFSCDMITEMNUMBER structures:

```
typedef struct _wfs_cdm_item_number
     {
     CHAR                       cCurrencyID[3];
     ULONG                      ulValues;
     USHORT                     usRelease;
     ULONG                      ulCount;
     USHORT                     usNumber;
     } WFSCDMITEMNUMBER, *LPWFSCDMITEMNUMBER;
```

cCurrencyID
A three character array storing the ISO format [Ref. 2] Currency ID; or three ASCII 0x20h characters if the currency of the item is not known.

ulValues
The value of a single item expressed in minimum dispense units; or a zero value if the value of the item is not known.

usRelease
The release of the item. The higher this number is, the newer the release. Zero means that there is only one release or the release is not known. This value has not been standardized and therefore a release number of the same item will not necessarily have the same value in different systems.

ulCount
The count of items of the same type moved to the same destination during the execution of this command.

usNumber
The logical number of the cash unit which received items during the execution of this command. This value will be zero if items were moved to the *usRetractArea* WFS_CDM_RA_TRANSPORT or WFS_CDM_RA_STACKER.

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_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.
WFS_ERR_CDM_UNSUPPOSITION	The output position specified is not supported.
WFS_ERR_CDM_POSITION_NOT_EMPTY	The retract area specified in *usRetractArea* is not empty so the retract operation is not possible.
WFS_ERR_CDM_INCOMPLETERETRACT	Some or all of the items were not retracted for a reason not covered by other error codes. The detail will be reported with the WFS_EXEE_CDM_INCOMPLETERETRACT event.

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 a retract or reject cash unit.
WFS_EXEE_CDM_CASHUNITERROR	An error occurred while attempting to retract to a cash unit.
WFS_SRVE_CDM_ITEMSTAKEN	The items presented have been removed by the user.
WFS_EXEE_CDM_INPUT_P6	Level 2 and/or level 3 notes have been detected.
WFS_EXEE_CDM_INFO_AVAILABLE	Information is available for items being processed by this operation.
WFS_EXEE_CDM_INCOMPLETERETRACT	The retract command has completed with an error and not all of the items have been retracted. The detail of what was actually retracted will be reported with the event data.
WFS_SRVE_CDM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.7 WFS_CMD_CDM_OPEN_SHUTTER

Description This command opens the shutter.

Input Param LPWORD lpfwPosition;

lpfwPosition
Pointer to the output position where the shutter is to be opened. If the application does not need to specify a shutter, this field can be set to NULL or its contents to WFS_CDM_POSNULL. The position 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 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_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.8 WFS_CMD_CDM_CLOSE_SHUTTER

Description This command closes the shutter.

Input Param LPWORD lpfwPosition;

lpfwPosition
Pointer to the output position where the shutter is to be closed. If the application does not need to specify a shutter, this field can be set to NULL or its contents to WFS_CDM_POSNULL. The position can 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 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_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.9 WFS_CMD_CDM_SET_TELLER_INFO

Description This command allows the application to set the teller position and initialize 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 structure 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 is not currently available.
WFS_ERR_CDM_INVALIDTELLERID	The teller ID is invalid. This error will never be generated by a Self-Service CDM.
WFS_ERR_CDM_UNSUPPOSITION	The position specified is not supported.
WFS_ERR_CDM_EXCHANGEACTIVE	The target teller is currently 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.

6.10 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
ulMinimum
ulMaximum
bAppLock
ulDispensedCount
ulPresentedCount
ulRetractedCount

As may the following fields of the WFSCDMPHCU structure:

ulInitialCount
ulCount
ulRejectCount
ulDispensedCount
ulPresentedCount
ulRetractedCount

Any other changes must be performed via an exchange operation.

If the fields *ulCount* and *ulRejectCount* of *lppPhysical* are set to zero 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. All cash units should be included not just the cash units whose values are to be 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. This error will never be generated by a Self-Service CDM.
WFS_ERR_CDM_INVALIDCASHUNIT	Invalid cash unit.
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM is in an exchange state.
WFS_ERR_CDM_CASHUNITERROR	A problem occurred with a cash unit. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.

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.
WFS_EXEE_CDM_CASHUNITERROR	An error occurred while accessing a cash unit.

Comments None.

6.11 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 all changes to a cash unit must take place while the cash unit is in an exchange state.

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_EXCHANGEACTIVE the CDM is in an exchange state.

While in an exchange state the CDM will process all WFS requests but exclude WFS[Async]Execute commands, except those listed below:

WFS_CMD_CDM_END_EXCHANGE

WFS_CMD_CDM_SET_MIX_TABLE

WFS_CMD_CDM_RESET

Any other WFS[Async]Execute commands will result in the error WFS_ERR_CDM_EXCHANGEACTIVE being generated.

If an error is returned by this command, the WFS_INF_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 can either be performed separately on each interface to the compound device, or the entire exchange operation can be done through the CIM interface.

Exchange via CDM and CIM interfaces

If the exchange is performed separately via the CDM and CIM interfaces then 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 (or multiple note identifiers associated with the same denomination) then the CIM interface should be used for exchange operations which affect these units. Those fields which are not common to both the CDM and CIM cash units are left unchanged when an exchange (or WFS_CMD_XXX_SET_CASH_UNIT_INFO) is executed on the other interface. For example if the CDM is used to set the current counts then the CIM *lpNoteNumberList* structure is not changed even if the data becomes inconsistent.

Exchange via the CIM Interface

All cash unit info fields exposed through the CDM interface are also exposed through the CIM interface, so the entire exchange operation for a recycling device can be achieved through the CIM interface.

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	A cash unit caused an error.
WFS_EXEE_CDM_NOTEERROR	An item detection error has occurred.
WFS_EXEE_CDM_INPUT_P6	Level 2 and/or level 3 notes have been detected.
WFS_EXEE_CDM_INFO_AVAILABLE	Information is available for items being processed by this operation.

Comments None.

6.12 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.

When *lpCUInfo* is not NULL 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 zero 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.

A WFS_EXEE_CDM_CASHUNITERROR event will be sent for any logical cash unit which cannot be successfully updated. If no cash units could be updated then a WFS_ERR_CDM_CASHUNITERROR code will be returned and WFS_EXEE_CDM_CASHUNITERROR events generated for every logical cash unit that could not be updated.

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. If this parameter is not NULL then it must contain the complete list of cash unit structures, not just the ones that have changed. If this parameter is NULL then any cash unit in a manipulated state (i.e. *usPStatus* value of WFS_CDM_STATCUMANIP) will remain in this state after the command completes.

The *usStatus* and *usPStatus* values passed in the cash unit structures included within the *lpCUInfo* parameter are ignored and the actual status of the cash units is determined when this command is executed. When *lpCUInfo* is not NULL and this command is successfully executed cash units will no longer be in a manipulated state (i.e. *usPStatus* will no longer be WFS_CDM_STATCUMANIP).

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	A cash unit problem occurred that meant no cash units could be updated. One or more WFS_EXEE_CDM_CASHUNITERROR events will be sent 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	A cash unit caused an error.
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.
WFS_EXEE_CDM_NOTEERROR	An item detection error has occurred.

WFS_EXEE_CDM_INPUT_P6	Level 2 and/or level 3 notes have been detected.
WFS_EXEE_CDM_INFO_AVAILABLE	Information is available for items being processed by this operation.

Comments None.

6.13 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.

6.14 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 a 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.

This parameter is a pointer to a pointer to WFSCDMITEMPOSITION structure.

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. When bills are presented to an output position this is the count of notes presented to the output position, any other notes rejected during the calibration process are not included in this count as they will be accounted for within the cash unit counts.

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.
WFS_ERR_CDM_INVALIDCASHUNIT	The cash unit number specified is not valid.

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.
WFS_EXEE_CDM_NOTEERROR	An item detection error has occurred.
WFS_EXEE_CDM_INPUT_P6	Level 2 and/or level 3 notes have been detected.
WFS_EXEE_CDM_INFO_AVAILABLE	Information is available for items being processed by this operation.

Comments None.

6.15 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.

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_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.

6.16 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 through WFS[Async]Lock on another application or service handle.

The device will attempt to move any items found anywhere within the device to the position specified within 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 will be generated and will inform the application where the items were actually moved to.

If an exchange state is active then this command will end the exchange state (even if this command does not complete successfully).

On a recycling device this command is not accepted if a cash-in transaction is active and will return a WFS_ERR_DEV_NOT_READY error.

If items are moved to a retract cash unit (i.e. a cash unit with *usType* WFS_CDM_TYPERETRACTCASSETTE), then the WFSCDMCASHUNIT.*ulCount* field of the retract cash unit must be incremented by 1 to specify the number of operations that changed the count. If items are moved to any other cash unit (e.g. a cash unit with *usType* WFS_CDM_TYPEREJECTCASSETTE), then the WFSCDMCASHUNIT.*ulCount* field of the cash unit must be incremented either by the number of items that were present at the time the WFS_CMD_CDM_RESET command was issued or the number counted by the device during the WFS_CMD_CDM_RESET command. Note that reject bin counts are unreliable.

Input Param

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

LPWFSCDMITEMPOSITION lpResetIn;

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

usNumber
If non-zero, this value specifies the *usNumber* (as specified by WFS_INF_CDM_CASH_UNIT_INFO) of the single cash unit to be used for the storage of any items found.

If items are to be moved to an output position, this value must be zero, *lpRetractArea* must be NULL and *fwOutputPosition* specifies where items are to be moved to.

If this value is zero and items are to be moved to internal areas of the device, *lpRetractArea* specifies where items are to be moved to or stored.

lpRetractArea
This field is used if items are to be moved to internal areas of the device, including cash units, the intermediate stacker, or the transport. The field is only relevant if *usNumber* is zero. The WFSCDMRETRACT structure is defined in WFS_CMD_CDM_RETRACT.

fwOutputPosition
This value will be ignored, because all items are moved from all positions.

usRetractArea
See the description in WFS_CMD_CDM_RETRACT.

usIndex
See the description in WFS_CMD_CDM_RETRACT.

fwOutputPosition
The output position to which items are to be moved. This field is only used if *usNumber* is zero and *lpRetractArea* is NULL. 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.

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. One or more WFS_EXEE_CDM_CASHUNITERROR events will be sent with the details.
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_INVALIDRETRACTPOSITION	The *usIndex* is not supported.
WFS_ERR_CDM_NOTRETRACTAREA	The retract area specified in *usRetractArea* is not supported.
WFS_ERR_CDM_POSITION_NOT_EMPTY	The retract area specified in *usRetractArea* is not empty so the moving of items was not possible.
WFS_ERR_CDM_INCOMPLETERETRACT	Some or all of the items were not retracted for a reason not covered by other error codes. The detail will be reported with the WFS_EXEE_CDM_INCOMPLETERETRACT event.

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_CASHUNITERROR	A cash unit caused an error.
WFS_SRVE_CDM_MEDIADETECTED	Media has been found in the device.
WFS_SRVE_CDM_ITEMSTAKEN	The items presented have been removed by the user.
WFS_EXEE_CDM_INPUT_P6	Level 2 and/or level 3 notes have been detected.
WFS_EXEE_CDM_INFO_AVAILABLE	Information is available for items being processed by this operation.
WFS_EXEE_CDM_INCOMPLETERETRACT	The command has completed with an error and not all of the items have been retracted. The detail of what was actually retracted will be reported in the WFS_EXEE_CDM_INCOMPLETERETRACT event data.
WFS_SRVE_CDM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.17 WFS_CMD_CDM_TEST_CASH_UNITS

Description This command is used to test cash units following replenishment. All physical cash units which are testable (i.e. that have a status of WFS_CDM_STATCUOK or WFS_CDM_STATCULOW and no application lock in the logical cash unit associated with the physical cash unit) are tested. 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 testable cash units regardless of the outcome of the test. This is the case if all testable cash units could be tested and a dispense was possible from at least one of the cash units.

A WFS_EXEE_CDM_CASHUNITERROR event will be sent for any logical cash unit which has one or more physical cash units which cannot be tested or which fail the test, even if the logical cash unit has other physical cash units which are successfully tested. If all the cash units could not be tested or no cash units are testable then a WFS_ERR_CDM_CASHUNITERROR code will be returned and WFS_EXEE_CDM_CASHUNITERROR events generated for every logical cash unit that encountered a problem. The operation performed to test the cash units is vendor dependent. Items may be dispensed or transported into a reject bin as a result of this command.

If no cash units are testable then a WFS_ERR_CDM_CASHUNITERROR code will be returned and WFS_EXEE_CDM_CASHUNITERROR events will be generated for every cash unit.

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 that meant all cash units could not be tested or no cash units were testable. One or more WFS_EXEE_CDM_CASHUNITERROR events 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 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_EXEE_CDM_CASHUNITERROR	A cash unit has failed the test or a cash unit was not testable.
WFS_SRVE_CDM_ITEMSTAKEN	The items presented have been removed by the user.
WFS_SRVE_CDM_CASHUNITINFOCHANGED	A cash unit was updated as a result of this command.
WFS_EXEE_CDM_NOTEERROR	An item detection error has occurred.
WFS_EXEE_CDM_INPUT_P6	Level 2 and/or level 3 notes have been detected.
WFS_SRVE_CDM_SHUTTERSTATUSCHANGED	The shutter status has changed.
WFS_EXEE_CDM_INFO_AVAILABLE	Information is available for items being processed by this operation.

Comments None.

6.18 WFS_CMD_CDM_SET_GUIDANCE_LIGHT

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

Input Param LPWFSCDMSETGUIDLIGHT lpSetGuidLight;

```
typedef struct _wfs_cdm_set_guidlight
     {
     WORD                      wGuidLight;
     DWORD                     dwCommand;
     } WFSCDMSETGUIDLIGHT, *LPWFSCDMSETGUIDLIGHT;
```

wGuidLight
Specifies the index of the guidance light to set as one of the values defined within the capabilities section in the *dwGuidLights [...]* field.

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

Value	Meaning	Type
WFS_CDM_GUIDANCE_OFF	The light indicator is turned off.	A
WFS_CDM_GUIDANCE_SLOW_FLASH	The light indicator is set to flash slowly.	B
WFS_CDM_GUIDANCE_MEDIUM_FLASH	The light indicator is set to flash medium frequency.	B
WFS_CDM_GUIDANCE_QUICK_FLASH	The light indicator is set to flash quickly.	B
WFS_CDM_GUIDANCE_CONTINUOUS	The light indicator is turned on continuously (steady).	B
WFS_CDM_GUIDANCE_RED	The light indicator color is set to red.	C
WFS_CDM_GUIDANCE_GREEN	The light indicator color is set to green.	C
WFS_CDM_GUIDANCE_YELLOW	The light indicator color is set to yellow.	C
WFS_CDM_GUIDANCE_BLUE	The light indicator color is set to blue.	C
WFS_CDM_GUIDANCE_CYAN	The light indicator color is set to cyan.	C
WFS_CDM_GUIDANCE_MAGENTA	The light indicator color is set to magenta.	C
WFS_CDM_GUIDANCE_WHITE	The light indicator color is set to white.	C
WFS_CDM_GUIDANCE_ENTRY	The light indicator is set to the entry state.	D
WFS_CDM_GUIDANCE_EXIT	The light indicator is set to the exit state.	D

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

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

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

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

6.19 WFS_CMD_CDM_POWER_SAVE_CONTROL

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

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

Input Param LPWFSCDMPOWERSAVECONTROL lpPowerSaveControl;

```
typedef struct _wfs_cdm_power_save_control
     {
     USHORT                    usMaxPowerSaveRecoveryTime;
     } WFSCDMPOWERSAVECONTROL, *LPWFSCDMPOWERSAVECONTROL;
```

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

Output Param None.

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

Value	Meaning
WFS_ERR_CDM_POWERSAVETOOSHORT	The power saving mode has not been activated because the device is not able to resume from the power saving mode within the specified *usMaxPowerSaveRecoveryTime* value.
WFS_ERR_CDM_POWERSAVEMEDIAPRESENT	The power saving mode has not been activated because media is present inside the device.
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_SRVE_CDM_POWER_SAVE_CHANGE	The power save recovery time has changed.

Comments None.

6.20 WFS_CMD_CDM_PREPARE_DISPENSE

Description On some hardware it can take a significant amount of time for the dispenser to get ready to dispense media. On this type of hardware the WFS_CMD_CDM_PREPARE_DISPENSE command can be used to improve transaction performance.

If this command is supported (see the *bPrepareDispense* capability) then applications can help to improve the time taken to dispense media by issuing this command as soon as the application knows that a dispense is likely to happen. This command either prepares the device for the next dispense operation, or terminates the dispense preparation if the subsequent dispense operation is no longer required.

With the exception of the WFS_CMD_CDM_DENOMINATE and WFS_CMD_CDM_DISPENSE commands, which will not stop the dispense preparation, any execute command on CDM or CIM will automatically stop the dispense preparation.

If this command is executed and the device is already in the specified *wAction s*tate, then this execution will have no effect and will complete with WFS_SUCCESS.

Input Param LPWFSCDMPREPAREDISPENSE lpPrepareDispense;

```
typedef struct _wfs_cdm_prepare_dispense
     {
     WORD                      wAction;
     } WFSCDMPREPAREDISPENSE, *LPWFSCDMPREPAREDISPENSE;
```

wAction
A value specifying the type of actions. The value is set to one of the following values:

Value	Meaning
WFS_CDM_START	Initiates the action to prepare for the next dispense command. This command does not wait until the device is ready to dispense before returning a completion event, it completes as soon as the preparation has been initiated.
WFS_CDM_STOP	Stops the previously activated dispense preparation. For example the motor of the transport will be stopped. This should be used if for some reason the subsequent dispense operation is no longer required.

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.

6.21 WFS_CMD_CDM_SET_BLACKLIST

Description This command is used to set all blacklist information. This list is persistent. Information set by this command overrides any existing blacklist or classification list, although it is not recommended that an application use both this command and WFS_CMD_CDM_SET_CLASSIFICATION_LIST to avoid overlap and confusion.

Input Param This parameter should be set to NULL if the application wishes to empty the blacklist.

LPWFSCDMBLACKLIST lpBlacklist;

The LPWFSCDMBLACKLIST structure is defined in the documentation of the WFS_INF_CDM_GET_BLACKLIST command.

lpszVersion
This is an application defined Unicode string that sets the version identifier of the blacklist. This can be set to NULL if it has no version identifier.

usCount
Number of pointers to WFSCDMBLACKLISTELEMENT structures returned in *lppBlacklistElements*.

lppBlacklistElements
Pointer to an array of pointers to WFSCDMBLACKLISTELEMENT structures. Each element represents a serial number, currency and value combination that a banknote will be matched against to determine if it is blacklisted.

The WFSCDMBLACKLISTELEMENT structure is defined in the documentation of the WFS_INF_CDM_GET_BLACKLIST command.

lpszSerialNumber
This Unicode string defines the serial number or a mask of serial numbers of one blacklist element with the defined currency and value. For a definition of the mask see Section 4.

cCurrencyID
The three character ISO format currency identifier [Ref. 2] of the blacklist element.

ulValue
The value of a blacklist element. This field can be set to zero to match all values.

Output Param None.

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

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

Comments Some classes of counterfeit banknotes have the same or similar serial numbers. By setting a serial number blacklist financial institutions can react quickly to a threat from counterfeit banknotes.

6.22 WFS_CMD_CDM_SYNCHRONIZE_COMMAND

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

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

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

Input Param LPWFSCDMSYNCHRONIZECOMMAND lpSynchronizeCommand;

```
typedef struct _wfs_cdm_synchronize_command
     {
     DWORD                    dwCommand;
     LPVOID                   lpCmdData;
     } WFSCDMSYNCHRONIZECOMMAND, *LPWFSCDMSYNCHRONIZECOMMAND;
```

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

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

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

Output Param None.

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

Value	Meaning
WFS_ERR_CDM_EXCHANGEACTIVE	The CDM is in an exchange state.
WFS_ERR_CDM_COMMANDUNSUPP	The command specified in the *dwCommand* field is not supported by the Service Provider.
WFS_ERR_CDM_SYNCHRONIZEUNSUPP	The preparation for the command specified in the *dwCommand* with the parameter specified in the *lpCmdData* is not supported by the Service Provider.

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

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

6.23 WFS_CMD_CDM_SET_CLASSIFICATION_LIST

Description This command is used to specify the entire note classification list. Any items not specified in this list will be handled according to normal classification rules. This information is persistent. Information set by this command overrides any existing blacklist or classification list, although it is not recommended that an application use both this command and WFS_CMD_CDM_SET_BLACKLIST to avoid overlap and confusion.

If a note is reclassified, it is handled as though it was a note of the new classification. For example, a fit note reclassified as unfit would be treated as though it were unfit, which may mean that the note is not dispensed.

Reclassification cannot be used to change a note's classification to a higher level, for example, a note recognized as counterfeit by the device cannot be reclassified as genuine. In addition, it is not possible to re-classify a level 2 note as level 1.

If two or more classification elements specify overlapping note definitions, but different *usLevel* values then the first one takes priority.

Input Param LPWFSCDMCLASSIFICATIONLIST lpClassificationList;

The LPWFSCDMCLASSIFICATIONLIST structure is defined in WFS_INF_CDM_GET_CLASSIFICATION_LIST. This parameter should be set to NULL if the application wishes to empty the note classification list.

Output Param None.

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

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

Comments None.

7. Events

7.1 WFS_SRVE_CDM_SAFEDOOROPEN

Description This service event is generated when the safe door has been opened.

Event Param None.

Comments None.

7.2 WFS_SRVE_CDM_SAFEDOORCLOSED

Description This service event is generated when the safe door has been closed.

Event Param None.

Comments None.

7.3 WFS_USRE_CDM_CASHUNITTHRESHOLD

Description This user event is generated when a threshold condition has occurred in one of the logical cash units. If the logical cash unit is a shared cash unit in a compound CIM/CDM then this event can also be generated as a result of a CIM operation.

This event can be triggered either by hardware sensors in the device or by the logical *ulCount* reaching the *ulMinimum* or *ulMaximum* value as specified in the WFSCDMCASHUNIT structure.

The application can check if the device has hardware sensors by querying the *bHardwareSensor* 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 will be triggered if the *ulMaximum* or *ulMinimum* values are not used and are set to zero.

In the situation where the cash unit is associated with multiple physical cash units the WFS_SRVE_CDM_CASHUNITINFOCHANGED event will 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 as well as the WFS_SRVE_CDM_CASHUNITINFOCHANGED event will be generated.

Event Param LPWFSCDMCASHUNIT lpCashUnit;

lpCashUnit
Pointer to a 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.

7.4 WFS_SRVE_CDM_CASHUNITINFOCHANGED

Description This service event is generated under the following circumstances:

- It is generated whenever *usStatus* and/or *usPStatus* changes. For instance, a physical cash unit has been removed or inserted, or a physical/logical cash unit has become empty or full.
- This event will also be generated for every cash unit changed in any way (including changes to counts, e.g. *ulCount, ulRejectCount, ulInitialCount, ulDispensedCount* and *ulPresentedCount*) as a result of the following commands:

  WFS_CMD_CDM_SET_CASH_UNIT_INFO
  WFS_CMD_CDM_END_EXCHANGE

- This event will also be fired when any change is made to a cash unit by the following commands, except for changes to counts (e.g. *ulCount, ulRejectCount, ulInitialCount, ulDispensedCount* and *ulPresentedCount*):

  WFS_CMD_CDM_CALIBRATE_CASH_UNIT
  WFS_CMD_CDM_TEST_CASH_UNITS

If the cash unit is shared cash unit in a compound CIM/CDM then this event can also be generated as a result of a CIM operation.

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

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

If a physical cash unit of a new logical type is inserted the cash unit structure reported by the last WFS_INF_CDM_CASH_UNIT_INFO command 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.

7.5 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.

7.6 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 an unsigned long holding the time in milliseconds by which the dispense operation will be delayed.

Comments None.

7.7 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.

7.8 WFS_EXEE_CDM_CASHUNITERROR

Description This execute event is generated if there is a problem with a cash unit during the execution of a command.

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 is invalid.
WFS_CDM_CASHUNITCONFIG	An attempt has been made to change the settings of a self-configuring cash unit.
WFS_CDM_CASHUNITNOTCONF	Specified cash unit is not configured.

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.

7.9 WFS_SRVE_CDM_ITEMSTAKEN

Description This service event is generated when items presented to the user have been taken. This event may be generated at any time.

Event Param LPWORD lpfwPosition;

lpfwPosition
Pointer to 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.

7.10 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 any CIM operation other than WFS_CMD_CIM_SET_CASH_UNIT_INFO and WFS_CMD_CIM_END_EXCHANGE.

Event Param LPWFSCDMCOUNTSCHANGED lpCountsChanged;

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

usCount
The size of *lpusCUNumList*.

lpusCUNumList
Pointer to a list of the *usNumber* values of the cash units whose counts have changed.

Comments None.

7.11 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
Pointer to an unsigned short holding the number of sub-dispense operations into which the dispense operation has been divided.

Comments None.

7.12 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.

7.13 WFS_EXEE_CDM_INCOMPLETEDISPENSE

Description This execute event is generated during WFS_CMD_CDM_DISPENSE when it has not been possible to dispense the entire denomination but part of the requested denomination is on the intermediate stacker or 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 are in customer access or on the intermediate stacker. WFS_INF_CDM_PRESENT_STATUS can be used to determine whether the items are in customer access.

Comments None.

7.14 WFS_EXEE_CDM_NOTEERROR

Description This execute event specifies the reason for a note detection error during the execution of a command.

Event Param LPUSHORT lpusReason;

lpusReason
Pointer to an unsigned short holding 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	An item counting error has occurred.
WFS_CDM_NOTESTOOCLOSE	Notes have been detected as being too close.
WFS_CDM_OTHERNOTEERROR	An item error not covered by the other values has been detected.
WFS_CDM_SHORTNOTEDETECTED	Short notes have been detected.

Comments None.

7.15 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.

7.16 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 section WFS_CMD_CDM_RESET. This parameter is a pointer to a pointer to WFSCDMITEMPOSITION structure.

Comments None.

7.17 WFS_EXEE_CDM_INPUT_P6

Description This execute event is generated if level 2 and/or level 3 notes are detected during execution of a CDM command.

Event Param None.

Comments None.

7.18 WFS_SRVE_CDM_DEVICEPOSITION

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

Event Param LPWFSCDMDEVICEPOSITION lpDevicePosition;

```
typedef struct _wfs_cdm_device_position
     {
     WORD                      wPosition;
     } WFSCDMDEVICEPOSITION, *LPWFSCDMDEVICEPOSITION;
```

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

Value	Meaning
WFS_CDM_DEVICEINPOSITION	The device is in its normal operating position.
WFS_CDM_DEVICENOTINPOSITION	The device has been removed from its normal operating position.
WFS_CDM_DEVICEPOSUNKNOWN	The position of the device cannot be determined.

Comments None.

7.19 WFS_SRVE_CDM_POWER_SAVE_CHANGE

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

Event Param LPWFSCDMPOWERSAVECHANGE lpPowerSaveChange;

```
typedef struct _wfs_cdm_power_save_change
     {
     USHORT               usPowerSaveRecoveryTime;
     } WFSCDMPOWERSAVECHANGE, *LPWFSCDMPOWERSAVECHANGE;
```

usPowerSaveRecoveryTime
Specifies the actual number of seconds required by the device to resume its normal operational state. This value is zero if the device exited the power saving mode.

Comments If another device class compounded with this device enters into a power saving mode, this device will automatically enter into the same power saving mode and this event will be generated.

7.20 WFS_EXEE_CDM_INFO_AVAILABLE

Description This execute event is generated when information is available for items being processed by the Service Provider.

Event Param LPWFSCDMITEMINFOSUMMARY *lppItemInfoSummary;

Pointer to a NULL-terminated array of pointers to WFSCDMITEMINFOSUMMARY structures, one structure for every level.

```
typedef struct _wfs_cdm_item_info_summary
     {
     USHORT              usLevel;
     USHORT              usNumOfItems;
     } WFSCDMITEMINFOSUMMARY, *LPWFSCDMITEMINFOSUMMARY;
```

usLevel
Defines the note level. Possible values are:

Value	Meaning
WFS_CDM_LEVEL_1	Information for level 1 notes.
WFS_CDM_LEVEL_2	Information for level 2 notes.
WFS_CDM_LEVEL_3	Information for level 3 notes.
WFS_CDM_LEVEL_4	Information for level 4 notes.

usNumOfItems
Number of items classified as *usLevel* which have information available.

Comments None.

7.21 WFS_EXEE_CDM_INCOMPLETERETRACT

Description This execute event is sent when the WFS_CMD_CDM_RETRACT or WFS_CMD_CDM_RESET command has completed with an error and not all of the items have been retracted.

Event Param LPWFSCDMINCOMPLETERETRACT lpIncompleteRetract;

```
typedef struct _wfs_cdm_incomplete_retract
     {
     WFSCDMITEMNUMBERLIST lpItemNumberList;
     USHORT               usReason;
     } WFSCDMINCOMPLETERETRACT, *LPWFSCDMINCOMPLETERETRACT;
```

lpItemNumberList
The *lpItemNumberList* parameter is a WFSCDMITEMNUMBERLIST structure (not a pointer) as defined in the description of the command WFS_CMD_CDM_RETRACT. Note that in this case the values in this structure report the amount and number of each denomination that were successfully moved during the command prior to the failure.

usReason
The reason for not having retracted items. The value is specified as one of the following values:

Value	Meaning
WFS_CDM_IRRETRACTFAILURE	The retract has partially failed for a reason not covered by the other reasons listed in this event, for example failing to pick an item to be retracted.
WFS_CDM_IRRETRACTAREAFULL	The specified retract area (see input parameter *usRetractArea*) has become full during the retract operation.
WFS_CDM_IRFOREIGNITEMSDETECTED	Foreign items have been detected.
WFS_CDM_IRINVALIDBUNCH	An invalid bunch of items has been detected, e.g. it is too large or could not be processed.

Comments None.

7.22 WFS_SRVE_CDM_SHUTTERSTATUSCHANGED

Description Within the limitations of the hardware sensors this service event is generated whenever the status of a shutter changes. The shutter status can change because of an explicit, implicit or manual operation depending on how the shutter is operated.

Event Param LPWFSCDMSHUTTERSTATUSCHANGED lpShutterStatusChanged;

```
typedef struct _wfs_cdm_shutter_status_changed
     {
     WORD                fwPosition;
     WORD                fwShutter;
     } WFSCDMSHUTTERSTATUSCHANGED, *LPWFSCDMSHUTTERSTATUSCHANGED;
```

fwPosition
Specifies one of the CDM output positions whose shutter status has changed 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
Specifies the new 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.

Comments None.

7.23 WFS_SRVE_CDM_ITEMSINSERTED

Description This service event is generated when items presented to the user have been re-inserted. This event may be generated at any time. This can be used to highlight to the customer that they may have dropped items back into the output position before the shutter is closed.

Event Param LPWORD lpfwPosition;

lpfwPosition
Pointer to the output position in which the items have been inserted. 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.

8. 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* field 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 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.			WFS_SRVE_CDM_SHUTTERSTATUSCHANGED(WFS_CDM_SHTOPEN) event generated	Items presented.
6.		WFS_CMD_CDM_DISPENSE completes successfully.		
7.	User takes bills.			
8.			WFS_SRVE_CDM_ITEMSTAKEN event generated. WFS_SRVE_CDM_SHUTTERSTATUSCHANGED(WFS_CDM_SHTCLOSED) 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_PARTIAL-DISPENSE event generated.	
5.			“Hardware level” dispense command issued for 50 USD.	
6.			WFS_SRVE_CDM_SHUTTERSTATUS CHANGED(WFS_CDM_SHTOPEN) event generated	Items presented.
7.			WFS_SRVE_CDM_SUBDISPENSEOK event generated.	
8.	User takes bills.			
9.			WFS_SRVE_CDM_ITEMSTAKEN event generated. WFS_SRVE_CDM_SHUTTERSTATUS CHANGED(WFS_CDM_SHTCLOSED) event generated	
10.			“Hardware level” dispense command issued for 50 USD.	
11.			WFS_SRVE_CDM_SHUTTERSTATUS CHANGED(WFS_CDM_SHTOPEN) event generated	Items presented.
12.			WFS_SRVE_CDM_SUBDISPENSE_O K event generated.	
13.	User takes bills.			
14.			WFS_SRVE_CDM_ITEMSTAKEN event generated. WFS_SRVE_CDM_SHUTTERSTATUS CHANGED(WFS_CDM_SHTCLOSED) event generated	
15.			“Hardware level” dispense command issued for 30 USD.	
16.			WFS_SRVE_CDM_SHUTTERSTATUS CHANGED(WFS_CDM_SHTOPEN) event generated	Items presented.
17.			WFS_SRVE_CDM_SUBDISPENSEOK event generated.	
18.		WFS_CMD_CDM_DISPENSE completes successfully.		
19.	User takes bills.			
20.			WFS_SRVE_CDM_ITEMSTAKEN event generated. WFS_SRVE_CDM_SHUTTERSTATUS CHANGED(WFS_CDM_SHTCLOSED) 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 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 command.	
4.			WFS_EXEE_CDM_PARTIAL-DISPENSE event generated.	
5.			“Hardware level” dispense command issued for 50 USD.	
6.			WFS_SRVE_CDM_SHUTTERSTATUS CHANGED(WFS_CDM_SHTOPEN) event generated	Items presented
7.			WFS_SRVE_CDM_SUBDISPENSEOK event generated.	
8.	User takes bills.			
9.			WFS_SRVE_CDM_ITEMSTAKEN event generated. WFS_SRVE_CDM_SHUTTERSTATUS CHANGED(WFS_CDM_SHTCLOSED) event generated	
10.			“Hardware level” dispense command issued for 50 USD.	
11.			WFS_SRVE_CDM_SHUTTERSTATUS CHANGED(WFS_CDM_SHTOPEN) event generated	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_ITEMSNOT-TAKEN.		

9. 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 and subsequently reinserted 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, reinserted 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.

10. Events Associated with Cash Unit Status Changes

The following instances illustrate which events will be posted when the cash unit statuses change. In all cases *bHardwareSensor* == TRUE, *ulMaximum* == 0 and *ulMinimum* == 0.

10.1 One Physical Cash Unit Goes LOW

The following table describes a dispense transaction case where the status of a physical cash unit only changes from WFS_CDM_STATCUOK to WFS_CDM_STATCULOW.

- *Logical CU 1 consists of Physical CU 1 and Physical CU 2*

	Action	**Status/Event**
1.		Logical CU 1: WFS_CDM_STATCUOK - Physical CU 1: WFS_CDM_STATCUOK - Physical CU 2: WFS_CDM_STATCUOK
2.	A user withdraws items.	
3.	The device dispenses and presents the items from Physical CU 1, whose status changes to low.	
4.	The status of Logical CU 1 does not change.	Logical CU 1: WFS_CDM_STATCUOK - Physical CU 1: **WFS_CDM_STATCULOW** - Physical CU 2: WFS_CDM_STATCUOK WFS_SRVE_CDM_CASHUNITINFOCHANGED

10.2 Last Physical Cash Unit Goes LOW

The following table describes a dispense transaction case where the status of a logical cash unit changes from WFS_CDM_STATCUOK to WFS_CDM_STATCULOW.

- *Logical CU 1 consists of Physical CU 1 and Physical CU 2*

	Action	**Status/Event**
1.		Logical CU 1: WFS_CDM_STATCUOK - Physical CU 1: WFS_CDM_STATCULOW - Physical CU 2: WFS_CDM_STATCUOK
2.	A user withdraws items.	
3.	The device dispenses and presents the items from Physical CU 2, whose status changes to low.	
4.	As a result, the status of Logical CU 1 changes to low.	Logical CU 1: **WFS_CDM_STATCULOW** - Physical CU 1: WFS_CDM_STATCULOW - Physical CU 2: **WFS_CDM_STATCULOW** WFS_SRVE_CDM_CASHUNITINFOCHANGED WFS_USRE_CDM_CASHUNITTHRESHOLD

10.3 One Physical Cash Unit Goes INOP

The following table describes a dispense transaction case where the status of a logical cash unit changes from WFS_CDM_STATCUOK to WFS_CDM_STATCULOW as the result of a physical cash unit failure.

- *Logical CU 1 consists of Physical CU 1 and Physical CU 2*
- *The device has ability to continue transaction when a problem occurs in a physical cash unit.*

	Action	**Status/Event**
1.		Logical CU 1: WFS_CDM_STATCUOK - Physical CU 1: WFS_CDM_STATCUOK - Physical CU 2: WFS_CDM_STATCULOW
2.	A user withdraws items.	
3.	The device tries to dispense the items from Physical CU 1; however, a problem occurs in the cash unit, whose status changes to inoperative.	
4.	The device complements the items by dispensing from Physical CU 2.	
5.	As a result, the status of Logical CU 1 changes to low.	Logical CU 1: **WFS_CDM_STATCULOW** - Physical CU 1: **WFS_CDM_STATCUINOP** - Physical CU 2: WFS_CDM_STATCULOW WFS_EXEE_CDM_CASHUNITERROR WFS_SRVE_CDM_CASHUNITINFOCHANGED WFS_USRE_CDM_CASHUNITTHRESHOLD

10.4 Last Physical Cash Unit Goes EMPTY

The following table describes a dispense transaction case where the status of a logical cash unit changes from WFS_CDM_STATCULOW to WFS_CDM_STATCUEMPTY.

- *Logical CU 1 consists of Physical CU 1 and Physical CU 2*

	Action	**Status/Event**
1.		Logical CU 1: WFS_CDM_STATCULOW - Physical CU 1: WFS_CDM_STATCUEMPTY - Physical CU 2: WFS_CDM_STATCULOW
2.	A user withdraws items.	
3.	The device dispenses and presents the items from Physical CU 2, whose status changes to empty.	
4.	As a result, the status of Logical CU 1 changes to empty.	Logical CU 1: WFS_CDM_STATCUEMPTY - Physical CU 1: WFS_CDM_STATCUEMPTY - Physical CU 2: WFS_CDM_STATCUEMPTY WFS_SRVE_CDM_CASHUNITINFOCHANGED

11. Multiple Dispense Command Flow

The Multiple Dispense feature occurs when a WFS_CMD_CDM_DISPENSE execute command is issued multiple times to place items on the stacker. Adding to the stacked items with a second or third dispense may be required where the initially picked Cash Unit fails to fulfill the full dispense request. The application can choose to add to the stacker from another Cash Unit to fulfill the request. Additionally this feature covers the requirement to dispense multiple currencies with a mix number other than WFS_CDM_INDIVIDUAL. Multiple currencies can be picked separately with the bunch of items assembled on the stacker before presentation to the customer.

Applications can refer to WFSCDMCAPS.*fwMoveItems* to determine if the Service Provider supports the feature. If a Service Provider supports this feature but an application does not wish to use it, the application should check WFSCDMSTATUS.*fwIntermediateStacker* to determine whether items on the stacker need to be purged prior to a dispense.

Multiple Dispense Example 'Out of Notes'

In the following example WFSCDMCAPS.*fwMoveItems* reports WFS_CDM_TOSTACKER. The CDM has 2 logical Cash Units. Cash Unit 1 has 20USD, Cash Unit 2 has 10USD. Cash Unit 1 has only 2 notes left.
This table illustrates multiple WFS_CMD_CDM_DISPENSE commands to fulfill a dispense request.

Step	End-User	Application	XFS SP	CDM Hardware
1.	User wants to dispense 60 USD.	WFS_CMD_CDM_DISPENSE *usMixNumber* = WFS_CDM_INDIVIDUAL *bPresent* = FALSE Request USD20x3 from Cash Unit 1.		2 of 3 items picked and stacked.
2.			WFS_EXEE_CDM_INCOMPLETEDISPENSE event. *lpDenomination* output records that 2x20USD were stacked.	
3.			WFS_CMD_CDM_DISPENSE completes WFS_ERR_CDM_NOTDISPENSABLE	
4.		The application decides that the dispense can be fulfilled and calculates the 20USD shortfall can be made up from the 10USD Cash Unit. WFS_CMD_CDM_DISPENSE *usMixNumber* = WFS_CDM_INDIVIDUAL *bPresent* = FALSE Request USD10x2 from Cash Unit 2.		2 of 2 notes picked and stacked together with the items already stacked.
5.			WFS_CMD_CDM_DISPENSE completes WFS_SUCCESS *lpDenomination* output records that 2x10USD were stacked.	
6.		Call WFS_CMD_CDM_PRESENT	WFS_SRVE_CDM_SHUTTERSTATUS-CHANGED(WFS_CDM_SHTOPEN) event generated	Items presented.
7.			WFS_CMD_CDM_PRESENT completes WFS_SUCCESS	
8.	User takes bills.		WFS_SRVE_CDM_ITEMSTAKEN event generated. WFS_SRVE_CDM_SHUTTERSTATUS-CHANGED(WFS_CDM_SHTCLOSED) event generated	

Multiple Dispense Example ‘Multiple Currency with Vendor Mix’

In the following example WFSCDMCAPS.*fwMoveItems* reports WFS_CDM_TOSTACKER.
The CDM has 2 logical Cash Units with good supply of notes in each. Cash Unit 1 has 20CHF, Cash Unit 2 has 20EUR.

This table illustrates multiple WFS_CMD_CDM_DISPENSE commands to fulfill a dispense request of EUR and CHF notes with vendor defined mix number. WFSCDMMIXTYPE.*usMixNumber* = 1.

Step	End-User	Application	XFS SP	CDM Hardware
1.	User wants to dispense 100 EUR and 100 CHF.	Application will split the dispense stacking all money before a single present. WFS_CMD_CDM_DISPENSE *usMixNumber* = 1 *bPresen*t = FALSE Request 100 EUR.		100 EUR picked and stacked.
2.			WFS_CMD_CDM_DISPENSE completes WFS_SUCCESS *lpDenomination* output records that 5x20EUR were stacked.	
3.		WFS_CMD_CDM_DISPENSE *usMixNumber* = WFS_CDM_INDIVIDUAL *bPresent* = FALSE Request 100 CHF.		100 CHF picked and stacked.
4.			WFS_CMD_CDM_DISPENSE completes WFS_SUCCESS *lpDenomination* output records that 5x20CHF were stacked.	
5.		Call WFS_CMD_CDM_PRESENT	WFS_SRVE_CDM_SHUTTERSTATUS CHANGED(WFS_CDM_SHTOPEN) event generated	Items presented.
6.			WFS_CMD_CDM_PRESENT completes WFS_SUCCESS	
7.	User takes bills.		WFS_SRVE_CDM_ITEMSTAKEN event generated. WFS_SRVE_CDM_SHUTTERSTATUS CHANGED(WFS_CDM_SHTCLOSED) event generated	

12. Appendix E – Cash Dispenser E2E Authentication

This section describes how to perform End to End authentication for a successful Dispense/Present process using the WFS_CMD_API_GET_COMMAND_NONCE, WFS_CMD_API_SECURE_COMMAND commands and WFS_INF_API_SECURE_QUERY queries. For a full description of these commands see section 13.2 of the API document.

An application can determine what functionality must be protected using E2E Authentication by calling the WFS_INF_API_SERVICE_INFO command and referencing the *lpE2EAuthentication* parameter. This points to a structure containing fields which detail the commands and queries that require authentication and also the timeout for the nonce obtained from the device. If no authentication is required, then *lpE2EAuthentication* is NULL.

Specified below is a description of the data that can be sent as part of a CDM Dispense process, and this is followed by flow diagrams showing how various use cases can be executed.

12.1 Secure Dispense Data Parameter Example Data

WFS_CMD_API_GET_COMMAND_NONCE

Input Parameters	**Data Format/Value**
lpszExtra	NULL
Output Parameters	
lpszNonce	The nonce from the device.
lpszExtra	Vendor specific data

WFS_CMD_API_SECURE_COMMAND

Input Parameters	**Data Format**
dwCommandID	WFS_CMD_CDM_DISPENSE
lpszInputToken	The formatted token that was constructed for the authentication calculation. This is a string of KEY=value pairs separated by commas. Example data: "NONCE=254611E63B2531576314E86527338D61,TOKENFORMAT=1,TOKENLENGTH=0164,DISPENSE1=50.00EUR,HMACSHA256=CB735612FD6141213C2827FB5A6A4F4846D7A7347B15434916FEA6AC16F3D2F2"
lpszResponseNonce	NULL
lpvInputData	Pointer to the WFSCDMDENOMINATION input structure for the dispense: Example: WFSCDMDENOMINATION.cCurrencyID[3] = "EUR"; WFSCDMDENOMINATION.ulAmount = 50; WFSCDMDENOMINATION.usCount = 4; WFSCDMDENOMINATION.lpulValues = [0021]; WFSCDMDENOMINATION.ulCashBox = 0;
Output Parameters	
dwCommandID	WFS_CMD_CDM_DISPENSE

lpszOutputToken	NULL
lpvOutputData	Points to an output structure for a Dispense, i.e. a WFSCDMDENOMINATION structure. Example: WFSCDMDENOMINATION.cCurrencyID[3] = “EUR”; WFSCDMDENOMINATION.ulAmount = 50; WFSCDMDENOMINATION.usCount = 4; WFSCDMDENOMINATION.lpulValues = [0021]; WFSCDMDENOMINATION.ulCashBox = 0;

WFS_INF_API_SECURE_QUERY

Input Parameters	**Data Format**
dwCategoryID	WFS_INF_CDM_PRESENT_STATUS
lpszInputToken	NULL
lpszResponseNonce	The nonce from the host.
lpvInputData	LPWORD lpfwPosition (the input parameter to WFS_INF_CDM_PRESENT_STATUS) Example: *lpfwPosition* = WFS_CDM_POSNULL
Output Parameters	
dwCategoryID	WFS_INF_CDM_PRESENT_STATUS
lpszOutputToken	The formatted token that was constructed for the authentication calculation. This is a string of KEY=value pairs separated by commas. This data also includes the nonce previously returned. Example data: "NONCE=1414,TOKENFORMAT=1,TOKENLENGTH=0268,DISPENSEID=CB735612FD6141213C2827FB5A6A4F4846D7A7347B15434916FEA6AC16F3D2F2,DISPENSED1=50.00EUR,PRESENTED1=YES,PRESENTEDAMOUNT1=50.00EUR,RETRACTED1=NO,HMACSHA256=55D123E9EE64F0CC3D1CD4F953348B441E521BBACCD6998C6F51D645D71E6C83"

lpvOutputData	LPWFSCDMPRESENTSTATUS lpPresentStatus (the output parameter to WFS_INF_CDM_PRESENT_STATUS)
	
	Example:
	WFSCDMDENOMINATION.cCurrencyID[3] = “EUR”;
	WFSCDMDENOMINATION.ulAmount = 50;
	WFSCDMDENOMINATION.usCount = 4;
	WFSCDMDENOMINATION.lpulValues = [0021];
	WFSCDMDENOMINATION.ulCashBox = 0;
	wPresentState = WFS_CDM_PRESENTED;
	lpszExtra = NULL;

Note: If end to end security is supported then the last present status values are **not** cleared if a dispense with an invalid token is received. If a dispense token is invalid the dispense will fail with an error to indicate that the token is invalid, and the WFS_INF_CDM_PRESENT_STATUS command will continue to report the existing status. This is to stop an attacker being able to reset the present status and conceal the last present result.

12.2 Secure Dispense Command Flow: - Dispense and Present

The following flow shows how the WFS_CMD_API_SECURE_COMMAND and WFS_INF_API_SECURE_QUERY commands are called to wrap the WFS_CMD_CDM_DISPENSE and WFS_INF_CDM_PRESENT_STATUS commands. The WFS_CMD_CDM_PRESENT command is used to present the bills to the user. This might be the case where the device has an intermediate stacker and requires a specific present command:

Application	CDM Commands and Events	API Commands and Events
Customer requests a cash operation and specifies an amount to be dispensed. Application calls WFS_CMD_API_GET_COMMAND_NONCE.		WFS_CMD_API_GET_COMMAND_NONCE called to initiate the Dispense End to End process.
		WFS_CMD_API_GET_COMMAND_NONCE completes successfully.
Application sends device nonce to the host. The host calculates the host MAC and responds with authorization, parameters, token, host MAC, and Host nonce.		
Application calls WFS_CMD_API_SECURE_COMMAND to execute the Dispense operation securely.		WFS_CMD_API_SECURE_COMMAND called, wrapping the WFS_CMD_CDM_DISPENSE command. On receiving this command the firmware verifies the data, then dispenses the cash to the intermediate stacker.
		WFS_CMD_API_SECURE_COMMAND completes with WFS_SUCCESS.
Application calls command to present the bills.	Calls WFS_CMD_CDM_PRESENT command. Once present has been called the existing device nonce is cleared.	WFS_SRVE_API_NONCE_CLEARED event generated.
	WFS_SRVE_CDM_SHUTTERSTATUS_CHANGED(WFS_CDM_SHTOPEN) event generated.	
	WFS_CMD_CDM_PRESENT completes WFS_SUCCESS.	
User takes bills.	WFS_SRVE_CDM_ITEMSTAKEN event generated. WFS_SRVE_CDM_SHUTTERSTATUS_CHANGED (WFS_CDM_SHTCLOSED) event generated.	
User now has the cash, so the application calls WFS_INF_API_SECURE_QUERY including the host nonce, to retrieve the device authenticated present status.		WFS_INF_API_SECURE_QUERY called, wrapping the WFS_INF_CDM_PRESENT_STATUS command

		WFS_INF_API_SECURE_QUERY completes with authenticated details of the End to End Dispense process.	
Application sends the authenticated details of what was dispensed to the host. The host verifies the authenticated details and updates the relevant account.			
End of process.			

12.3 Secure Dispense Command Flow – Dispense Only with no Present

The following flow shows how the WFS_CMD_API_SECURE_COMMAND and WFS_INF_API_SECURE_QUERY commands are called to wrap the WFS_CMD_CDM_DISPENSE and WFS_INF_CDM_PRESENT_STATUS commands. The WFS_CMD_CDM_PRESENT command is not used to present the bills to the user. This would be the use case where the device has no intermediate stacker and does not require a present command:

Application	CDM Commands and Events	API Commands and Events
Customer requests a cash operation and specifies an amount to be dispensed. Application calls WFS_CMD_API_GET_COMMAND_NONCE.		WFS_CMD_API_GET_COMMAND_NONCE called to initiate the Dispense End to End process.
		WFS_CMD_API_GET_COMMAND_NONCE completes successfully.
Application sends device nonce to the host. The host calculates the host MAC and responds with authorization, parameters, token, Host MAC, and Host nonce.		
Application calls WFS_CMD_API_SECURE_COMMAND to execute the Dispense operation securely.		WFS_CMD_API_SECURE_COMMAND called, wrapping the WFS_CMD_CDM_DISPENSE command. On receiving this command, the firmware verifies the data, then dispenses the cash to the output position.
		WFS_SRVE_API_NONCE_CLEARED event generated.
		WFS_CMD_API_SECURE_COMMAND completes with WFS_SUCCESS.
User now has the cash, so the application calls WFS_INF_API_SECURE_QUERY including the host nonce, to retrieve the device authenticated present status.		WFS_INF_API_SECURE_QUERY called, wrapping the WFS_INF_CDM_PRESENT_STATUS command. The existing device nonce is cleared.
		WFS_INF_API_SECURE_QUERY completes with authenticated details of the End to End Dispense process.
Application sends the authenticated details of what was dispensed to the host. The host verifies the authenticated details and updates the relevant account.		
End of process.		

12.4 Secure Dispense Command Flow: - Dispense Completes With Error Followed by an Additional Dispense and Present

The following flow shows how the WFS_CMD_API_SECURE_COMMAND and WFS_INF_API_SECURE_QUERY commands are called to wrap the WFS_CMD_CDM_DISPENSE and WFS_INF_CDM_PRESENT_STATUS commands. The WFS_CMD_CDM_PRESENT command is used to present the bills to the user.
In this use case the WFS_CMD_API_SECURE_COMMAND wraps the Dispense command which subsequently fails with a WFS_ERR_CDM_CASHUNITERROR error code and the notes moved to the reject bin automatically. The application then decides to retry the Dispense and this is successful. The WFS_CMD_CDM_PRESENT command is then called to present the bills to the user. The token describing what was dispensed with an HMAC calculated over it is then retrieved using the WFS_INF_API_SECURE_QUERY command, wrapping the WFS_INF_CDM_PRESENT_STATUS command. The result is then sent to the host.

Application	CDM Commands and Events	API Commands and Events
Customer requests a cash operation and specifies an amount to be dispensed. Application calls WFS_CMD_API_GET_COMMAND_NONCE.		WFS_CMD_API_GET_COMMAND_NONCE called to initiate the Dispense End to End process.
		WFS_CMD_API_GET_COMMAND_NONCE completes successfully.
Application sends device nonce to the host. The host calculates the host MAC and responds with authorization, parameters, token, Host MAC, and Host nonce.		
Application calls WFS_CMD_API_SECURE_COMMAND to execute the Dispense operation securely.		WFS_CMD_API_SECURE_COMMAND called, wrapping the WFS_CMD_CDM_DISPENSE command. On receiving this command, the firmware verifies the data then dispenses the cash to the intermediate stacker.
	WFS_EXEE_CDM_CASHUNITERROR event.	WFS_CMD_API_SECURE_COMMAND completes with the WFS_ERR_CDM_CASHUNITERROR error code. Dispensed cash is automatically moved to the reject bin, and the intermediate stacker is empty.
The application decides that the dispense can be retried for the same amount. The existing token is still valid at this point and can be reused since no cash has been presented.		WFS_CMD_API_SECURE_COMMAND called again, wrapping the WFS_CMD_CDM_DISPENSE command. On receiving this command, the firmware verifies the data then dispenses the cash to the intermediate stacker.
		WFS_CMD_API_SECURE_COMMAND completes with WFS_SUCCESS.
Application calls command to present the bills.	Calls WFS_CMD_CDM_PRESENT command. Once present has been called the existing device nonce is cleared.	

		WFS_SRVE_API_NONCE_CLEARED event generated.
	WFS_SRVE_CDM_SHUTTERSTATUS_CHANGED(WFS_CDM_SHTOPEN) event generated.	
	WFS_CMD_CDM_PRESENT command completes with WFS_SUCCESS.	
User takes bills.	WFS_SRVE_CDM_ITEMSTAKEN event generated. WFS_SRVE_CDM_SHUTTERSTATUS_CHANGED (WFS_CDM_SHTCLOSED) event generated.	
User now has the cash, so the application calls WFS_INF_API_SECURE_QUERY including the host nonce, to retrieve the device authenticated present status.		WFS_INF_API_SECURE_QUERY called, wrapping the WFS_INF_CDM_PRESENT_STATUS command.
		WFS_INF_API_SECURE_QUERY completes with authenticated details of the End to End Dispense process.
Application sends the authenticated details of what was dispensed to the host. The host verifies the authenticated details and updates the relevant account.		
End of process.		

12.5 Secure Dispense Command Flow: - User does not remove bills. Dispense, Present and Retract

The following flow shows how the WFS_CMD_API_SECURE_COMMAND and WFS_INF_API_SECURE_QUERY commands are called to wrap the WFS_CMD_CDM_DISPENSE and WFS_INF_CDM_PRESENT_STATUS commands. The WFS_CMD_CDM_PRESENT command is used to present the bills to the user.
In this case the user does not remove all of the bills, therefore the bills are retracted using the WFS_CMD_CDM_RETRACT command and a message is sent to the host to report it.

Application	CDM Commands and Events	API Commands and Events
Customer requests a cash operation and specifies an amount to be dispensed. Application calls WFS_CMD_API_GET_COMMAND_NONCE.		WFS_CMD_API_GET_COMMAND_NONCE called to initiate the Dispense End to End process.
		WFS_CMD_API_GET_COMMAND_NONCE completes successfully.
Application sends device nonce to the host. The host calculates the host MAC and responds with authorization, parameters, token, Host MAC, and Host nonce.		
Application calls WFS_CMD_API_SECURE_COMMAND to execute the Dispense operation securely.		WFS_CMD_API_SECURE_COMMAND called, wrapping the WFS_CMD_CDM_DISPENSE command. On receiving this command the firmware verifies the data, then dispenses the cash to the intermediate stacker.
		WFS_CMD_API_SECURE_COMMAND completes with WFS_SUCCESS.
Application calls command to present the bills.	Calls WFS_CMD_CDM_PRESENT command. Once present has been called the existing device nonce is cleared.	
		WFS_SRVE_API_NONCE_CLEARED event generated. usReason = WFS_API_NONCECLR_COMPLETED.
	WFS_SRVE_CDM_SHUTTERSTATUS_CHANGED(WFS_CDM_SHTOPEN) event generated.	
	WFS_CMD_CDM_PRESENT command completes with WFS_SUCCESS.	
User does not remove bills, so the application issues a Retract command.	Calls WFS_CMD_CDM_RETRACT command.	
User takes bills.	WFS_SRVE_CDM_SHUTTERSTATUS_CHANGED (WFS_CDM_SHTCLOSED) event generated.	

	Calls WFS_CMD_CDM_RETRACT command completes with WFS_SUCCESS.	
The application calls WFS_INF_API_SECURE_QUERY including the host nonce, to retrieve the device authenticated present status.		WFS_INF_API_SECURE_QUERY called, wrapping the WFS_INF_CDM_PRESENT_STATUS command.
		WFS_INF_API_SECURE_QUERY completes with authenticated details of the End to End Dispense process.
Application sends the authenticated details of what was dispensed and retracted to the host.		
End of process.		

12.6 Secure Dispense Command Flow: - Authentication Process Timeout

The following flow shows the WFS_API_CMD_SECURE_COMMAND dispensing bills, but no Present command is called. In this case the authenticated process times out from the device.

Application	CDM Commands and Events	API Commands and Events
Customer requests a cash operation and specifies an amount to be dispensed. Application calls WFS_CMD_API_SECURE_PROCESS_START.		WFS_CMD_API_GET_COMMAND_NONCE called to initiate the Dispense End to End process.
		WFS_CMD_API_GET_COMMAND_NONCE completes successfully.
Application sends Dispense request token and key name to the host. The host calculates the host MAC and responds with authorization, parameters, token and Host MAC.		
Application calls WFS_CMD_API_SECURE_COMMAND to begin the Dispense process.		WFS_CMD_API_SECURE_COMMAND called. On receiving this command the firmware verifies the data, then dispenses the cash to the output position.
		WFS_CMD_API_SECURE_COMMAND completes with WFS_SUCCESS.
The application fails to call the Present command, and the device times out waiting for one. An event is sent to inform the application.		WFS_SRVE_API_NONCE_CLEARED event generated. usReason = WFS_API_NONCECLR_EXPIRED.
Application clears the stacker.	Calls WFS_CMD_CDM_REJECT command.	
	WFS_CMD_CDM_REJECT command completes with WFS_SUCCESS.	
End of process.		

~~12.~~13. C - Header file

```
/******************************************************************************
*                                                                             *
* xfscdm.h      XFS - Cash Dispenser (CDM) definitions                        *
*                                                                             *
*               Version 3.~~40  (December 6 2019)~~50  (November 18 2022)
*
*                                                                             *
******************************************************************************/

#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           (~~0x2803~~0x3203) /* Version 3.~~40~~50
*/
#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)
#define     WFS_INF_CDM_GET_ITEM_INFO               (CDM_SERVICE_OFFSET + 10)
#define     WFS_INF_CDM_GET_BLACKLIST               (CDM_SERVICE_OFFSET + 11)
#define     WFS_INF_CDM_GET_ALL_ITEMS_INFO          (CDM_SERVICE_OFFSET + 12)
#define     WFS_INF_CDM_GET_CLASSIFICATION_LIST     (CDM_SERVICE_OFFSET + 13)


/* 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)
#define     WFS_CMD_CDM_SET_GUIDANCE_LIGHT          (CDM_SERVICE_OFFSET + 24)
```

```
#define     WFS_CMD_CDM_POWER_SAVE_CONTROL          (CDM_SERVICE_OFFSET + 25)
#define     WFS_CMD_CDM_PREPARE_DISPENSE            (CDM_SERVICE_OFFSET + 26)
#define     WFS_CMD_CDM_SET_BLACKLIST               (CDM_SERVICE_OFFSET + 27)
#define     WFS_CMD_CDM_SYNCHRONIZE_COMMAND         (CDM_SERVICE_OFFSET + 28)
#define     WFS_CMD_CDM_SET_CLASSIFICATION_LIST     (CDM_SERVICE_OFFSET + 29)


/* 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_SRVE_CDM_MEDIADETECTED              (CDM_SERVICE_OFFSET + 17)
#define     WFS_EXEE_CDM_INPUT_P6                   (CDM_SERVICE_OFFSET + 18)
#define     WFS_SRVE_CDM_DEVICEPOSITION             (CDM_SERVICE_OFFSET + 19)
#define     WFS_SRVE_CDM_POWER_SAVE_CHANGE          (CDM_SERVICE_OFFSET + 20)
#define     WFS_EXEE_CDM_INFO_AVAILABLE             (CDM_SERVICE_OFFSET + 21)
#define     WFS_EXEE_CDM_INCOMPLETERETRACT          (CDM_SERVICE_OFFSET + 22)
#define     WFS_SRVE_CDM_SHUTTERSTATUSCHANGED       (CDM_SERVICE_OFFSET + 23)
#define     WFS_SRVE_CDM_ITEMSINSERTED              (CDM_SERVICE_OFFSET + 24)

/* 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
#define     WFS_CDM_DEVFRAUDATTEMPT                 WFS_STAT_DEVFRAUDATTEMPT
#define     WFS_CDM_DEVPOTENTIALFRAUD               WFS_STAT_DEVPOTENTIALFRAUD

/* 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)

/* Size and max index of dwGuidLights array */
```

```
#define     WFS_CDM_GUIDLIGHTS_SIZE                 (32)
#define     WFS_CDM_GUIDLIGHTS_MAX                  (WFS_CDM_GUIDLIGHTS_SIZE - 1)

/* Indices of WFSCDMSTATUS.dwGuidLights [...]
             WFSCDMCAPS.dwGuidLights [...] */

#define     WFS_CDM_GUIDANCE_POSOUTNULL             (0)
#define     WFS_CDM_GUIDANCE_POSOUTLEFT             (1)
#define     WFS_CDM_GUIDANCE_POSOUTRIGHT            (2)
#define     WFS_CDM_GUIDANCE_POSOUTCENTER           (3)
#define     WFS_CDM_GUIDANCE_POSOUTTOP              (4)
#define     WFS_CDM_GUIDANCE_POSOUTBOTTOM           (5)
#define     WFS_CDM_GUIDANCE_POSOUTFRONT            (6)
#define     WFS_CDM_GUIDANCE_POSOUTREAR             (7)

/* Values of WFSCDMSTATUS.dwGuidLights [...]
             WFSCDMCAPS.dwGuidLights [...] */

#define     WFS_CDM_GUIDANCE_NOT_AVAILABLE          (0x00000000)
#define     WFS_CDM_GUIDANCE_OFF                    (0x00000001)
#define     WFS_CDM_GUIDANCE_SLOW_FLASH             (0x00000004)
#define     WFS_CDM_GUIDANCE_MEDIUM_FLASH           (0x00000008)
#define     WFS_CDM_GUIDANCE_QUICK_FLASH            (0x00000010)
#define     WFS_CDM_GUIDANCE_CONTINUOUS             (0x00000080)
#define     WFS_CDM_GUIDANCE_RED                    (0x00000100)
#define     WFS_CDM_GUIDANCE_GREEN                  (0x00000200)
#define     WFS_CDM_GUIDANCE_YELLOW                 (0x00000400)
#define     WFS_CDM_GUIDANCE_BLUE                   (0x00000800)
#define     WFS_CDM_GUIDANCE_CYAN                   (0x00001000)
#define     WFS_CDM_GUIDANCE_MAGENTA                (0x00002000)
#define     WFS_CDM_GUIDANCE_WHITE                  (0x00004000)
#define     WFS_CDM_GUIDANCE_ENTRY                  (0x00100000)
#define     WFS_CDM_GUIDANCE_EXIT                   (0x00200000)

/* values of WFSCDMSTATUS.wDevicePosition
             WFSCDMDEVICEPOSITION.wPosition */

#define     WFS_CDM_DEVICEINPOSITION                (0)
#define     WFS_CDM_DEVICENOTINPOSITION             (1)
#define     WFS_CDM_DEVICEPOSUNKNOWN                (2)
#define     WFS_CDM_DEVICEPOSNOTSUPP                (3)

/* 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 WFSCDMOUTPOS.fwJammedShutterPosition */

#define     WFS_CDM_SHUTTERPOS_NOTSUPPORTED           (0)
#define     WFS_CDM_SHUTTERPOS_NOTJAMMED              (1)
#define     WFS_CDM_SHUTTERPOS_OPEN                   (2)
#define     WFS_CDM_SHUTTERPOS_PARTIALLY_OPEN         (3)
#define     WFS_CDM_SHUTTERPOS_CLOSED                 (4)
#define     WFS_CDM_SHUTTERPOS_UNKNOWN                (5)

/* 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,
             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)
#define     WFS_CDM_RA_ITEMCASSETTE                   (0x0020)

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

#define     WFS_CDM_PRESENT                           (0x0001)
#define     WFS_CDM_RETRACT                           (0x0002)
#define     WFS_CDM_REJECT                            (0x0004)
#define     WFS_CDM_NOTSUPP                           (0x0008)
#define     WFS_CDM_ITEMCASSETTE                      (0x0010)

/* values of WFSCDMCAPS.fwMoveItems */

#define     WFS_CDM_FROMCU                            (0x0001)
#define     WFS_CDM_TOCU                              (0x0002)
#define     WFS_CDM_TOTRANSPORT                       (0x0004)
#define     WFS_CDM_TOSTACKER                         (0x0008)

/* 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,
             WFSCDMCAPS.fwPositions,
             WFSCDMOUTPOS.fwPosition,
             WFSCDMTELLERDETAILS.fwOutputPosition,
             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_POSFRONT                        (0x0800)
#define     WFS_CDM_POSREAR                         (0x1000)

/* additional values of WFSCDMPHYSICALCU.fwPosition */
#define     WFS_CDM_POSREJECT                       (0x0100)

/* 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)
#define     WFS_CDM_CASHUNITNOTCONF                 (8)
```

```
/* values of lpusReason in WFS_EXEE_CDM_NOTEERROR */

#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)
#define     WFS_CDM_OTHERNOTEERROR                    (6)
#define     WFS_CDM_SHORTNOTEDETECTED                 (7)

/* values of WFSCDMPREPAREDISPENSE.wAction */

#define     WFS_CDM_START                             (1)
#define     WFS_CDM_STOP                              (2)

/* values of WFSCDMSTATUS.wAntiFraudModule */

#define     WFS_CDM_AFMNOTSUPP                        (0)
#define     WFS_CDM_AFMOK                             (1)
#define     WFS_CDM_AFMINOP                           (2)
#define     WFS_CDM_AFMDEVICEDETECTED                 (3)
#define     WFS_CDM_AFMUNKNOWN                        (4)

/* values of WFSCDMGETITEMINFO.usLevel,
             WFSCDMITEMINFOSUMMARY.usLevel,
             WFSCDMGETALLITEMSINFO.usLevel,
             WFSCDMITEMINFOALL.usLevel */

#define     WFS_CDM_LEVEL_1                     (1)
#define     WFS_CDM_LEVEL_2                     (2)
#define     WFS_CDM_LEVEL_3                     (3)
#define     WFS_CDM_LEVEL_4                     (4)

/* values of WFSCDMITEMINFOALL.usLevel */

#define     WFS_CDM_LEVEL_ALL                   (0)

/* values for WFSCDMGETITEMINFO.dwItemInfoType */

#define     WFS_CDM_ITEM_SERIALNUMBER           (0x00000001)
#define     WFS_CDM_ITEM_SIGNATURE              (0x00000002)
#define     WFS_CDM_ITEM_IMAGEFILE              (0x00000004)

/* values of lpusReason in WFS_EXEE_CDM_INCOMPLETERETRACT */

#define     WFS_CDM_IRRETRACTFAILURE                  (1)
#define     WFS_CDM_IRRETRACTAREAFULL                 (2)
#define     WFS_CDM_IRFOREIGNITEMSDETECTED            (3)
#define     WFS_CDM_IRINVALIDBUNCH                    (4)

/* values for WFSCDMITEMINFOALL.wOnBlacklist */

#define     WFS_CDM_ONBLACKLIST                 (0x0001)
#define     WFS_CDM_NOTONBLACKLIST              (0x0002)
#define     WFS_CDM_BLACKLISTUNKNOWN            (0x0003)

/* values for WFSCDMITEMINFOALL.wItemLocation */

#define     WFS_CDM_LOCATION_DEVICE             (0x0001)
#define     WFS_CDM_LOCATION_CASHUNIT           (0x0002)
#define     WFS_CDM_LOCATION_CUSTOMER           (0x0003)
#define     WFS_CDM_LOCATION_UNKNOWN            (0x0004)

/* values for WFSCDMITEMINFOALL.wOnClassificationList */

#define     WFS_CDM_CLASSIFICATIONLIST_ON       (0x0001)
#define     WFS_CDM_CLASSIFICATIONLIST_NOTON    (0x0002)
#define     WFS_CDM_CLASSIFICATIONLIST_UNKNOWN  (0x0003)

/* values for WFSCDMITEMINFOALL.wItemDeviceLocation */
```

```
#define     WFS_CDM_DEVLOC_STACKER              (0x0001)
#define     WFS_CDM_DEVLOC_OUTPUT               (0x0002)
#define     WFS_CDM_DEVLOC_TRANSPORT            (0x0003)
#define     WFS_CDM_DEVLOC_UNKNOWN              (0x0004)

/* 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))
#define WFS_ERR_CDM_INVALID_PORT                      (-(CDM_SERVICE_OFFSET + 37))
#define WFS_ERR_CDM_POWERSAVETOOSHORT                 (-(CDM_SERVICE_OFFSET + 38))
#define WFS_ERR_CDM_POWERSAVEMEDIAPRESENT             (-(CDM_SERVICE_OFFSET + 39))
#define WFS_ERR_CDM_POSITION_NOT_EMPTY                (-(CDM_SERVICE_OFFSET + 40))
#define WFS_ERR_CDM_INCOMPLETERETRACT                 (-(CDM_SERVICE_OFFSET + 41))
#define WFS_ERR_CDM_COMMANDUNSUPP                     (-(CDM_SERVICE_OFFSET + 42))
#define WFS_ERR_CDM_SYNCHRONIZEUNSUPP                 (-(CDM_SERVICE_OFFSET + 43))

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

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

typedef struct _wfs_cdm_status
{
    WORD                  fwDevice;
    WORD                  fwSafeDoor;
    WORD                  fwDispenser;
    WORD                  fwIntermediateStacker;
    LPWFSCDMOUTPOS        *lppPositions;
    LPSTR                 lpszExtra;
    DWORD                 dwGuidLights[WFS_CDM_GUIDLIGHTS_SIZE];
    WORD                  wDevicePosition;
```

```
    USHORT                  usPowerSaveRecoveryTime;
    WORD                    wAntiFraudModule;
} 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;
    DWORD                   dwGuidLights[WFS_CDM_GUIDLIGHTS_SIZE];
    BOOL                    bPowerSaveControl;
    BOOL                    bPrepareDispense;
    BOOL                    bAntiFraudModule;
    DWORD                   dwItemInfoTypes;
    BOOL                    bBlacklist;
    LPDWORD                 lpdwSynchronizableCommands;
    BOOL                    bClassificationList;
} WFSCDMCAPS, *LPWFSCDMCAPS;

typedef struct _wfs_cdm_physicalcu
{
    LPSTR                   lpPhysicalPositionName;
    CHAR                    cUnitID[5];
    ULONG                   ulInitialCount;
    ULONG                   ulCount;
    ULONG                   ulRejectCount;
    ULONG                   ulMaximum;
    USHORT                  usPStatus;
    BOOL                    bHardwareSensor;
    ULONG                   ulDispensedCount;
    ULONG                   ulPresentedCount;
    ULONG                   ulRetractedCount;
} 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;
    ULONG                   ulDispensedCount;
    ULONG                   ulPresentedCount;
    ULONG                   ulRetractedCount;
} 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;

typedef struct _wfs_cdm_signature
{
    ULONG                 ulLength;
    LPVOID                lpData;
} WFSCDMSIGNATURE, *LPWFSCDMSIGNATURE;

typedef struct _wfs_cdm_get_item_info
{
    USHORT                usLevel;
    USHORT                usIndex;
    DWORD                 dwItemInfoType;
} WFSCDMGETITEMINFO, *LPWFSCDMGETITEMINFO;

typedef struct _wfs_cdm_item_info
{
    CHAR                  cCurrencyID[3];
    ULONG                 ulValue;
    USHORT                usRelease;
    LPWSTR                lpszSerialNumber;
    LPWFSCDMSIGNATURE     lpSignature;
    LPSTR                 lpszImageFileName;
} WFSCDMITEMINFO, *LPWFSCDMITEMINFO;

typedef struct _wfs_cdm_get_all_items_info
{
    USHORT                usLevel;
} WFSCDMGETALLITEMSINFO, *LPWFSCDMGETALLITEMSINFO;

typedef struct _wfs_cdm_item_info_all
{
    USHORT                usLevel;
    CHAR                  cCurrencyID[3];
    ULONG                 ulValue;
    USHORT                usRelease;
    LPWSTR                lpszSerialNumber;
    LPSTR                 lpszImageFileName;
    WORD                  wOnBlacklist;
    WORD                  wItemLocation;
    USHORT                usNumber;
    WORD                  wOnClassificationList;
    WORD                  wItemDeviceLocation;
} WFSCDMITEMINFOALL, *LPWFSCDMITEMINFOALL;

typedef struct _wfs_cdm_all_items_info
{
    USHORT                usCount;
    LPWFSCDMITEMINFOALL   *lppItemsList;
} WFSCDMALLITEMSINFO, *LPWFSCDMALLITEMSINFO;

typedef struct _wfs_cdm_classification_element
{
    LPWSTR                lpszSerialNumber;
    CHAR                  cCurrencyID[3];
    ULONG                 ulValue;
    USHORT                usLevel;
    BOOL                  bUnfit;
} WFSCDMCLASSIFICATIONELEMENT, *LPWFSCDMCLASSIFICATIONELEMENT;

typedef struct _wfs_cdm_classification_list
{
    LPWSTR                lpszVersion;
    USHORT                usCount;
```

```
    LPWFSCDMCLASSIFICATIONELEMENT *lppClassificationElements;
} WFSCDMCLASSIFICATIONLIST, *LPWFSCDMCLASSIFICATIONLIST;

/*=============================================================*/
/* 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_item_number
{
    CHAR                 cCurrencyID[3];
    ULONG                ulValues;
    USHORT               usRelease;
    ULONG                ulCount;
    USHORT               usNumber;
} WFSCDMITEMNUMBER, *LPWFSCDMITEMNUMBER;

typedef struct _wfs_cdm_item_number_list
{
    USHORT               usNumOfItemNumbers;
    LPWFSCDMITEMNUMBER    *lppItemNumber;
} WFSCDMITEMNUMBERLIST, *LPWFSCDMITEMNUMBERLIST;

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;

typedef struct _wfs_cdm_set_guidlight
{
    WORD                  wGuidLight;
    DWORD                 dwCommand;
} WFSCDMSETGUIDLIGHT, *LPWFSCDMSETGUIDLIGHT;

typedef struct _wfs_cdm_power_save_control
{
    USHORT                usMaxPowerSaveRecoveryTime;
} WFSCDMPOWERSAVECONTROL, *LPWFSCDMPOWERSAVECONTROL;

typedef struct _wfs_cdm_prepare_dispense
{
    WORD                  wAction;
} WFSCDMPREPAREDISPENSE, *LPWFSCDMPREPAREDISPENSE;

typedef struct _wfs_cdm_blacklist_element
{
    LPWSTR                lpszSerialNumber;
    CHAR                  cCurrencyID[3];
    ULONG                 ulValue;
} WFSCDMBLACKLISTELEMENT, *LPWFSCDMBLACKLISTELEMENT;

typedef struct _wfs_cdm_blacklist
{
    LPWSTR                lpszVersion;
    USHORT                usCount;
    LPWFSCDMBLACKLISTELEMENT *lppBlacklistElements;
} WFSCDMBLACKLIST, *LPWFSCDMBLACKLIST;

typedef struct _wfs_cdm_synchronize_command
{
    DWORD                 dwCommand;
    LPVOID                lpCmdData;
} WFSCDMSYNCHRONIZECOMMAND, *LPWFSCDMSYNCHRONIZECOMMAND;

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

typedef struct _wfs_cdm_cu_error
{
    WORD                  wFailure;
    LPWFSCDMCASHUNIT      lpCashUnit;
```

```
} WFSCDMCUERROR, *LPWFSCDMCUERROR;

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

typedef struct _wfs_cdm_device_position
{
    WORD                  wPosition;
} WFSCDMDEVICEPOSITION, *LPWFSCDMDEVICEPOSITION;

typedef struct _wfs_cdm_power_save_change
{
    USHORT                usPowerSaveRecoveryTime;
} WFSCDMPOWERSAVECHANGE, *LPWFSCDMPOWERSAVECHANGE;

typedef struct _wfs_cdm_item_info_summary
{
    USHORT                usLevel;
    USHORT                usNumOfItems;
} WFSCDMITEMINFOSUMMARY, *LPWFSCDMITEMINFOSUMMARY;

typedef struct _wfs_cdm_incomplete_retract
{
    WFSCDMITEMNUMBERLIST  lpItemNumberList;
    USHORT                usReason;
} WFSCDMINCOMPLETERETRACT, *LPWFSCDMINCOMPLETERETRACT;

typedef struct _wfs_cdm_shutter_status_changed
{
    WORD                  fwPosition;
    WORD                  fwShutter;
} WFSCDMSHUTTERSTATUSCHANGED, *LPWFSCDMSHUTTERSTATUSCHANGED;

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

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

#endif  /* __INC_XFSCDM__H */
```