Extensions for Financial Services (XFS) interface specification
Release 3.20 - Part 5: Cash Dispenser Device Class Interface
Programmer’s Reference

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties, the constitution of which is indicated in the foreword of this Workshop Agreement.

The formal process followed by the Workshop in the development of this Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN-CENELEC Management Centre can be held accountable for the technical content of this CEN Workshop Agreement or possible conflicts with standards or legislation.

This CEN Workshop Agreement can in no way be held as being an official standard developed by CEN and its Members.

This CEN Workshop Agreement is publicly available as a reference document from the CEN Members National Standard Bodies.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and United Kingdom.
# Table of Contents

Foreword ....................................................................................................................... 4

1. Introduction .......................................................................................................... 7  
   1.1 Background to Release 3.20 ........................................................................... 7  
   1.2 XFS Service-Specific Programming .................................................................. 7  

2. Cash Dispensers ................................................................................................. 8  

3. References ........................................................................................................... 9  

4. Info Commands ................................................................................................. 10  
   4.1 WFS_INF_CDM_STATUS ................................................................................... 10  
   4.2 WFS_INF_CDM_CAPABILITIES ...................................................................... 15  
   4.3 WFS_INF_CDM_CASH_UNIT_INFO .................................................................. 19  
   4.4 WFS_INF_CDM_TELLER_INFO ........................................................................ 26  
   4.5 WFS_INF_CDM_CURRENCY_EXP ..................................................................... 28  
   4.6 WFS_INF_CDM_MIX_TYPES .......................................................................... 29  
   4.7 WFS_INF_CDM_MIX_TABLE ............................................................................ 30  
   4.8 WFS_INF_CDM_PRESENT_STATUS ................................................................. 31  

5. Execute Commands .......................................................................................... 33  
   5.1 WFS_CMD_CDM_DENOMINATE ...................................................................... 33  
   5.2 WFS_CMD_CDM_DISPENSE .......................................................................... 36  
   5.3 WFS_CMD_CDM_COUNT ................................................................................. 39  
   5.4 WFS_CMD_CDM_PRESENT .............................................................................. 42  
   5.5 WFS_CMD_CDM_REJECT ............................................................................... 43  
   5.6 WFS_CMD_CDM_RETRACT ............................................................................. 44  
   5.7 WFS_CMD_CDM_OPEN_SHUTTER .................................................................... 47  
   5.8 WFS_CMD_CDM_CLOSE_SHUTTER ................................................................... 48  
   5.9 WFS_CMD_CDM_SET_TELLER_INFO ............................................................... 49  
   5.10 WFS_CMD_CDM_SET_CASH_UNIT_INFO ...................................................... 50  
   5.11 WFS_CMD_CDM_START_EXCHANGE ............................................................ 52  
   5.12 WFS_CMD_CDM_END_EXCHANGE ............................................................... 54  
   5.13 WFS_CMD_CDM_OPEN_SAFE_DOOR ............................................................. 56  
   5.14 WFS_CMD_CDM_CALIBRATE_CASH_UNIT .................................................. 57  
   5.15 WFS_CMD_CDM_SET_MIX_TABLE ............................................................... 59  
   5.16 WFS_CMD_CDM_RESET .............................................................................. 60  
   5.17 WFS_CMD_CDM_TEST_CASH_UNITS ......................................................... 62  
   5.18 WFS_CMD_CDM_SET_GUIDANCE_LIGHT .................................................. 64  
   5.19 WFS_CMD_CDM_POWER_SAVE_CONTROL ............................................... 66  
   5.20 WFS_CMD_CDM_PREPARE_DISPENSE ...................................................... 67
6. Events ............................................................................................................................... 68
   6.1 WFS_SRVE_CDM_SAFEDOOROPEN .............................................................................. 68
   6.2 WFS_SRVE_CDM_SAFEDOORCLOSED ......................................................................... 69
   6.3 WFS_USRE_CDM_CASHUNITTHRESHOLD ................................................................. 70
   6.4 WFS_SRVE_CDM_CASHUNITINFOCHANGED ......................................................... 71
   6.5 WFS_SRVE_CDM_TELLERINFOCHANGED ............................................................... 72
   6.6 WFS_EXEE_CDM_DELAYEDDISPENSE ....................................................................... 73
   6.7 WFS_EXEE_CDM_STARTDISPENSE ............................................................................ 74
   6.8 WFS_EXEE_CDM_CASHUNITERROR .......................................................................... 75
   6.9 WFS_SRVE_CDM_ITEMSTAKEN ................................................................................. 76
   6.10 WFS_SRVE_CDM_COUNTS_CHANGED ....................................................................... 77
   6.11 WFS_EXEE_CDM_PARTIALDISPENSE ....................................................................... 78
   6.12 WFS_EXEE_CDM_SUBDISPENSEOK ......................................................................... 79
   6.13 WFS_EXEE_CDM_INCOMPLETEDISPOSE .................................................................. 80
   6.14 WFS_EXEE_CDM_NOTEERROR .................................................................................. 81
   6.15 WFS_SRVE_CDM_ITEMSPRESENTED ....................................................................... 82
   6.16 WFS_SRVE_CDM_MEDIADETECTED ......................................................................... 83
   6.17 WFS_EXEE_CDM_INPUT_P6 ..................................................................................... 84
   6.18 WFS_SRVE_CDM_DEVICEPOSITION ........................................................................ 85
   6.19 WFS_SRVE_CDM_POWER_SAVE_CHANGE ................................................................ 86

7. Sub-Dispensing Command Flow ...................................................................................... 87

8. Rules for Cash Unit Exchange ....................................................................................... 90

9. C - Header file ............................................................................................................... 91
Foreword

This CWA is revision 3.20 of the XFS interface specification.

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties on 2011-06-29, the constitution of which was supported by CEN following the public call for participation made on 1998-06-24. The specification is continuously reviewed and commented in the CEN/ISSS Workshop on XFS. It is therefore expected that an update of the specification will be published in due time as a CWA, superseding this revision 3.20.

A list of the individuals and organizations which supported the technical consensus represented by the CEN Workshop Agreement is available to purchasers from the CEN-CENELEC Management Centre. These organizations were drawn from the banking sector. The CEN/ISSS XFS Workshop gathered suppliers as well as banks and other financial service companies.

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

Part 1: Application Programming Interface (API) - Service Provider Interface (SPI) - Programmer's Reference
Part 2: Service Classes Definition - Programmer's Reference
Part 3: Printer and Scanning Device Class Interface Programmer's Reference
Part 4: Identification Card Device Class Interface - Programmer's Reference
Part 5: Cash Dispenser Device Class Interface - Programmer's Reference
Part 6: PIN Keypad Device Class Interface - Programmer's Reference
Part 7: Check Reader/Scanner Device Class Interface - Programmer's Reference
Part 8: Depository Device Class Interface - Programmer's Reference
Part 9: Text Terminal Unit Device Class Interface - Programmer's Reference
Part 10: Sensors and Indicators Unit Device Class Interface - Programmer's Reference
Part 11: Vendor Dependent Mode Device Class Interface - Programmer's Reference
Part 12: Camera Device Class Interface - Programmer's Reference
Part 13: Alarm Device Class Interface - Programmer's Reference
Part 14: Card Embossing Unit Class Interface - Programmer's Reference
Part 15: Cash-In Module Device Class Interface - Programmer's Reference
Part 16: Card Dispenser Device Class Interface - Programmer's Reference
Part 17: Barcode Reader Device Class Interface - Programmer's Reference
Part 18: Item Processing Module Device Class Interface - Programmer's Reference
Parts 19 - 28: Reserved for future use.

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

Part 29: XFS MIB Architecture and SNMP Extensions
Part 30: XFS MIB Device Specific Definitions - Printer Device Class
Part 31: XFS MIB Device Specific Definitions - Identification Card Device Class
Part 32: XFS MIB Device Specific Definitions - Cash Dispenser Device Class
Part 33: XFS MIB Device Specific Definitions - PIN Keypad Device Class
Part 34: XFS MIB Device Specific Definitions - Check Reader/Scanner Device Class
Part 35: XFS MIB Device Specific Definitions - Depository Device Class
Part 36: XFS MIB Device Specific Definitions - Text Terminal Unit Device Class
Part 37: XFS MIB Device Specific Definitions - Sensors and Indicators Unit Device Class
Part 38: XFS MIB Device Specific Definitions - Camera Device Class
Part 39: XFS MIB Device Specific Definitions - Alarm Device Class
Part 40: XFS MIB Device Specific Definitions - Card Embossing Unit Device Class
Part 41: XFS MIB Device Specific Definitions - Cash-In Module Device Class
Part 42: Reserved for future use.
Part 43: XFS MIB Device Specific Definitions - Vendor Dependent Mode Class
Part 44: XFS MIB Application Management
Part 45: XFS MIB Device Specific Definitions - Card Dispenser Device Class
Part 46: XFS MIB Device Specific Definitions - Barcode Reader Device Class
Part 47: XFS MIB Device Specific Definitions - Item Processing Module Device Class
Parts 48 - 60 are reserved for future use.
Part 61: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 62: Printer and Scanning Device Class Interface - Migration from Version 3.10 (CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 63: Identification Card Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 65: PIN Keypad Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 67: Depository Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 71: Camera Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 72: Alarm Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.10 (CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 74: Cash-In Module Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 75: Card Dispenser Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 76: Barcode Reader Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 77: Item Processing Module Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
In addition to these Programmer's Reference specifications, the reader of this CWA is also referred to a complementary document, called Release Notes. The Release Notes contain clarifications and explanations on the CWA specifications, which are not requiring functional changes. The current version of the Release Notes is available online from http://www.cen.eu/cen/pages/default.aspx.
The information in this document represents the Workshop's current views on the issues discussed as of the date of publication. It is furnished for informational purposes only and is subject to change without notice. CEN/ISSS makes no warranty, express or implied, with respect to this document.

The formal process followed by the Workshop in the development of the CEN Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN-CENELEC Management Centre can be held accountable for the technical content of the CEN Workshop Agreement or possible conflict with standards or legislation. This CEN Workshop Agreement can in no way be held as being an official standard developed by CEN and its members.

The final review/endorsement round for this CWA was started on 2011-06-23 and was successfully closed on 2011-07-23. The final text of this CWA was submitted to CEN for publication on 2011-08-26.

This CEN Workshop Agreement is publicly available as a reference document from the National Members of CEN: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and the United Kingdom.

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

Revision History:

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

1.1 Background to Release 3.20

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

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

Release 3.20 of the XFS specification is based on a C API and is delivered with the continued promise for the protection of technical investment for existing applications. This release of the specification extends the functionality and capabilities of the existing devices covered by the specification, but does not include any new device classes. Notable major enhancements include Mixed Media processing to allow mixed cash and check accepting, as well as the addition of new commands to the CIM, PTR and IDC to allow better support of the Japanese marketplace.

1.2 XFS Service-Specific Programming

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

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

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

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

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

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

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

This design allows implementation of applications that can be used with a range of services that provide differing subsets of the functionalities that are defined for their service class. Applications may use the WFSGetInfo and WFSAsyncGetInfo commands to inquire about the capabilities of the service they are about to use, and modify their behavior accordingly, or they may use functions and then deal with WFS_ERR_UNSUPP_COMMAND error returns to make decisions as to how to use the service.
2. 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
### References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer’s Reference, Revision 3.20
2. ISO 4217 at http://www.iso.org
3. XFS Cash-In Module Device Class Interface, Programmer’s Reference, Revision 3.20
4. Info Commands

4.1 WFS_INF_CDM_STATUS

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

Input Param
None.

Output Param
LPWFS_CDMSTATUS lpStatus;

typedef struct _wfs_cdm_status
{
  WORD     fwDevice;
  WORD     fwSafeDoor;
  WORD     fwDispenser;
  WORD     fwIntermediateStacker;
  LPWFS_CDMOUTPOS   *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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_DEVONLINE</td>
<td>The device is online. This is returned when the dispenser is present and operational.</td>
</tr>
<tr>
<td>WFS_CDM_DEVOFFLINE</td>
<td>The device is offline (e.g. the operator has taken the device offline by turning a switch or pulling out the device).</td>
</tr>
<tr>
<td>WFS_CDM_DEVPOWEROFF</td>
<td>The device is powered off or physically not connected.</td>
</tr>
<tr>
<td>WFS_CDM_DEVNODEVICE</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_CDM_DEVHWERROR</td>
<td>The device is inoperable due to a hardware error.</td>
</tr>
<tr>
<td>WFS_CDM_DEVUSERERROR</td>
<td>The device is present but a person is preventing proper device operation.</td>
</tr>
<tr>
<td>WFS_CDM_DEVBUSY</td>
<td>The device is busy and unable to process an execute command at this time.</td>
</tr>
<tr>
<td>WFS_CDM_DEVFRAUDATTEMPT</td>
<td>The device is present but is inoperable because it has detected a fraud attempt.</td>
</tr>
<tr>
<td>WFS_CDM_DEVPOTENTIALFRAUD</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_DOORNOT_SUPPORTED</td>
<td>Physical device has no safe door or safe door state reporting is not supported.</td>
</tr>
<tr>
<td>WFS_CDM_DOOROPEN</td>
<td>Safe door is open.</td>
</tr>
<tr>
<td>WFS_CDM_DOORCLOSED</td>
<td>Safe door is closed.</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_DISPOK</td>
<td>All cash units present are in a good state.</td>
</tr>
<tr>
<td>WFS_CDM_DISPCUSTATE</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_CDM_DISPCUSTOP</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_CDM_DISPCUUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the cash units cannot be determined.</td>
</tr>
</tbody>
</table>

/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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_ISEMPTY</td>
<td>The intermediate stacker is empty.</td>
</tr>
<tr>
<td>WFS_CDM_ISNOTEMPTY</td>
<td>The intermediate stacker is not empty. The items have not been in customer access.</td>
</tr>
<tr>
<td>WFS_CDM_ISNOTEMPTYCUST</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_CDM_ISNOTEMPTYUNK</td>
<td>The intermediate stacker is not empty. It is not known if the items have been in customer access.</td>
</tr>
<tr>
<td>WFS_CDM_ISUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the intermediate stacker cannot be determined.</td>
</tr>
<tr>
<td>WFS_CDM_ISNOTSUPPORTED</td>
<td>The physical device has no intermediate stacker.</td>
</tr>
</tbody>
</table>

/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:

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

/fwPosition
Supplies the output position as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_POSLEFT</td>
<td>Left output position.</td>
</tr>
</tbody>
</table>
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:

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

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_PSEMPTY</td>
<td>The output position is empty.</td>
</tr>
<tr>
<td>WFS_CDM_PSNOTEMPTY</td>
<td>The output position is not empty.</td>
</tr>
<tr>
<td>WFS_CDM_PSUNKOWN</td>
<td>Due to a hardware error or other condition, the state of the output position cannot be determined.</td>
</tr>
<tr>
<td>WFS_CDM_PSNOTSUPPORTED</td>
<td>The device is not capable of reporting whether or not items are at the output position.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_TPOK</td>
<td>The transport is in a good state.</td>
</tr>
<tr>
<td>WFS_CDM_TPINOP</td>
<td>The transport is inoperative due to a hardware failure or media jam.</td>
</tr>
<tr>
<td>WFS_CDM_TPKUNKOWN</td>
<td>Due to a hardware error or other condition the state of the transport cannot be determined.</td>
</tr>
<tr>
<td>WFS_CDM_TPNOTSUPPORTED</td>
<td>The physical device has no transport or transport state reporting is not supported.</td>
</tr>
</tbody>
</table>

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_TPTSTATEMPTY</td>
<td>The transport is empty.</td>
</tr>
<tr>
<td>WFS_CDM_TPSTATNOTEMPTY</td>
<td>The transport is not empty.</td>
</tr>
<tr>
<td>WFS_CDM_TPSTATNOTEMPTYCUST</td>
<td>Items which a customer has had access to are on the transport.</td>
</tr>
<tr>
<td>WFS_CDM_TPSTATNOTEMPTY_UNK</td>
<td>Due to a hardware error or other condition it is not known whether there are items on the transport.</td>
</tr>
<tr>
<td>WFS_CDM_TPSTATNOTSUPPORTED</td>
<td>The device is not capable of reporting whether items are on the transport.</td>
</tr>
</tbody>
</table>
lpszExtra
Pointer to a list of vendor-specific, or any other extended, information. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.

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, and optionally one type C.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_GUIDANCE_NOT_AVAILABLE</td>
<td>The status is not available.</td>
<td>A</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_OFF</td>
<td>The light is turned off.</td>
<td>A</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_SLOW_FLASH</td>
<td>The light is blinking slowly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_MEDIUM_FLASH</td>
<td>The light is blinking medium frequency.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_QUICK_FLASH</td>
<td>The light is blinking quickly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_CONTINUOUS</td>
<td>The light is turned on continuous (steady).</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_RED</td>
<td>The light is red.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_GREEN</td>
<td>The light is green.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_YELLOW</td>
<td>The light is yellow.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_BLUE</td>
<td>The light is blue.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_CYAN</td>
<td>The light is cyan.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_MAGENTA</td>
<td>The light is magenta.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_WHITE</td>
<td>The light is white.</td>
<td>C</td>
</tr>
</tbody>
</table>

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:

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_AFMNOTSUPP</td>
<td>No anti-fraud module is available.</td>
</tr>
<tr>
<td>WFS_CDM_AFMOK</td>
<td>Anti-fraud module is in a good state and no foreign device is detected.</td>
</tr>
</tbody>
</table>
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.
4.2 WFS_INF_CDM_CAPABILITIES

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

Input Param
None.

Output Param
LPWFSCDMCAPS lpCaps;

typedef struct _wfs_cdm_caps
{
    WORD     wClass;
    WORD     fwType;
    WORD     wMaxDispenseItems;
    BOOL     bCompound;
    BOOL     bShutter;
    BOOL     bShutterControl;
    WORD     fwRetractAreas;
    WORD     fwRetractTransportActions;
    WORD     fwRetractStackerActions;
    BOOL     bSafeDoor;
    BOOL     bCashBox;
    BOOL     bIntermediateStacker;
    BOOL     bItemsTakenSensor;
    WORD     fwPositions;
    WORD     fwMoveItems;
    WORD     fwExchangeType;
    LPSTR     lpszExtra;
    DWORD     dwGuidLights[WFS_CDM_GUIDLIGHTS_SIZE];
    BOOL     bPowerSaveControl;
    BOOL     bPrepareDispense;
    BOOL     bAntiFraudModule;
} 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_TELLERBILL</td>
<td>The CDM is a Teller Bill Dispenser.</td>
</tr>
<tr>
<td>WFS_CDM_SELFSERVICEBILL</td>
<td>The CDM is a Self-Service Bill Dispenser.</td>
</tr>
<tr>
<td>WFS_CDM_TELLERCOIN</td>
<td>The CDM is a Teller Coin Dispenser.</td>
</tr>
<tr>
<td>WFS_CDM_SELFSERVICECOIN</td>
<td>The CDM is a Self-Service Coin Dispenser.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_RA_RETRACT</td>
<td>The items may be retracted to a retract cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_RA_TRANSPORT</td>
<td>The items may be retracted to the transport.</td>
</tr>
<tr>
<td>WFS_CDM_RA_STACKER</td>
<td>The items may be retracted to the intermediate stacker.</td>
</tr>
<tr>
<td>WFS_CDM_RA_REJECT</td>
<td>The items may be retracted to a reject cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_RA_ITEMCASSETTE</td>
<td>The items may be retracted to the item cassettes, i.e. cassettes that can be dispensed from.</td>
</tr>
</tbody>
</table>

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 or from the transport this value will be WFS_CDM_NOTSUPP. This field will be a combination of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_PRESENT</td>
<td>The items may be presented.</td>
</tr>
<tr>
<td>WFS_CDM_RETRACT</td>
<td>The items may be retracted to a retract cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_REJECT</td>
<td>The items may be retracted to a reject bin.</td>
</tr>
<tr>
<td>WFS_CDM_ITEMCASSETTE</td>
<td>The items may be retracted to the item cassettes, i.e. cassettes that can be dispensed from.</td>
</tr>
</tbody>
</table>

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 or from the stacker this value will be WFS_CDM_NOTSUPP. Otherwise it will be a combination of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_PRESENT</td>
<td>The items may be presented.</td>
</tr>
<tr>
<td>WFS_CDM_RETRACT</td>
<td>The items may be retracted to a retract cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_REJECT</td>
<td>The items may be retracted to a reject bin.</td>
</tr>
<tr>
<td>WFS_CDM_ITEMCASSETTE</td>
<td>The items may be retracted to the item cassettes, i.e. cassettes that can be dispensed from.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_POSLEFT</td>
<td>The CDM has a left output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSRIGHT</td>
<td>The CDM has a right output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSCENTER</td>
<td>The CDM has a center output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSTOP</td>
<td>The CDM has a top output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSBOTTOM</td>
<td>The CDM has a bottom output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSFRONT</td>
<td>The CDM has a front output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSREAR</td>
<td>The CDM has a rear output position.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_FROMCU</td>
<td>The CDM can move items from the cash units to the intermediate stacker while there are items on the transport.</td>
</tr>
<tr>
<td>WFS_CDM_TOCU</td>
<td>The CDM can retract items to the cash units while there are items on the intermediate stacker.</td>
</tr>
<tr>
<td>WFS_CDM_TOTRANSPORT</td>
<td>The CDM can retract items to the transport while there are items on the intermediate stacker.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_EXBYHAND</td>
<td>The CDM supports manual replenishment either by filling the cash unit by hand or by replacing the cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_EXTOCASSETTES</td>
<td>The CDM supports moving items from the replenishment cash unit to another cash unit.</td>
</tr>
</tbody>
</table>

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

**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.

The elements of this array are specified as a combination of the following flags and indicate all of the possible flash rates (type B) and colors (type C) that the guidance light indicator is capable of handling. If the guidance light indicator only supports one color then no value of type C is returned. A value of WFS_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.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_GUIDANCE_NOTAVAILABLE</td>
<td>There is no guidance light control available at this position.</td>
<td>A</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_OFF</td>
<td>The light can be off.</td>
<td>B</td>
</tr>
</tbody>
</table>
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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_GUIDANCE_POSOUTNULL</td>
<td>The default output position.</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_POSOUTLEFT</td>
<td>Left output position.</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_POSOUTRIGHT</td>
<td>Right output position.</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_POSOUTCENTER</td>
<td>Center output position.</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_POSOUTTOP</td>
<td>Top output position.</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_POSOUTBOTTOM</td>
<td>Bottom output position.</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_POSOUTFRONT</td>
<td>Front output position.</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_POSOUTREAR</td>
<td>Rear output position.</td>
</tr>
</tbody>
</table>

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.

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.
4.3 WFS_INF_CDM_CASH_UNIT_INFO

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

Where a logical cash unit is configured but there is no corresponding physical cash unit currently present in the device, information about the missing cash unit will still be returned in the lp CashUnitInfo 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 lp CashUnitInfo 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, lp CashUnitInfo will contain information about the three logical cash units and a us Count 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 lp CashUnitInfo 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:

- lpsz CashUnitName
- us Type
- cCurrencyID
- ul Values

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 (us Number), i.e. us Number 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 ul Count (on logical and physical cash units) reaches zero it will not decrement further but will remain at zero. When ul Count 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 ul Count 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 ul Count reaches zero then WFS_CDMIDGETEMPTY should not be generated when ul Count 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 ul Count is zero if the value in ul Count 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 ul Count reaches zero, then the number of notes in the cash unit can be determined relative to ul InitialCount using ul DispensedCount, ul RetractedCount and the CIM ul CashInCount, e.g. Number of Notes = ul InitialCount − ul DispensedCount + ul RetractedCount + CIM::ul CashInCount.

Threshold Events
The threshold event WFS USRE_CDM_CASHUNITTHRESHOLD can be triggered either by hardware sensors in the device or by the ul Count reaching the ul Minimum or ul Maximum value.
The application can check if the device has this capability 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 can be triggered.

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

**Exchanges**

If a physical cash unit is 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_TYPEREEJECTCASSETTE` / `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**

```c
LPWFS_CDMCUINFO lpCashUnitInfo;
typedef struct _wfs_cdm_cu_info
{
    USHORT           usTellerID;
    USHORT           usCount;
    LPWFS_CDM_CASHUNIT *lppList;
} WFS_CDMCUINFO, *LPWFS_CDMCUINFO;
```

**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 `WFS_CDM_CASHUNIT` 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_TYPENA</td>
<td>Not applicable. Typically means cash unit is missing.</td>
</tr>
<tr>
<td>WFS_CDM_TYPEREJECTCASSETTE</td>
<td>Reject cash unit. This type will also indicate a combined reject/retract cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_TYPEBILLCASSETTE</td>
<td>Cash unit containing bills.</td>
</tr>
<tr>
<td>WFS_CDM_TYPECOINCYLINDER</td>
<td>Coin cylinder.</td>
</tr>
<tr>
<td>WFS_CDM_TYPECOINDISPENSER</td>
<td>Coin dispenser as a whole unit.</td>
</tr>
<tr>
<td>WFS_CDM_TYPERETRACTCASSETTE</td>
<td>Retract cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_TYPECOUPON</td>
<td>Cash unit containing coupons or advertising material.</td>
</tr>
<tr>
<td>WFS_CDM_TYPEDOCUMENT</td>
<td>Cash unit containing documents.</td>
</tr>
<tr>
<td>WFS_CDM_TYPEREPCONTAINER</td>
<td>Replenishment container. A cash unit can be refilled from a replenishment container.</td>
</tr>
<tr>
<td>WFS_CDM_TYPERECYCLING</td>
<td>Recycling cash unit. This unit is only present when the device is a compound device with a CIM.</td>
</tr>
</tbody>
</table>

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_TYPETYPECOINCYLINDER, WFS_CDM_TYPETYPECOINDISPENSER, WFS_CDM_TYPETYPECOUPON, WFS_CDM_TYPETYPEDOCUMENT or WFS_CDM_TYPETYPECYCLING), 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_TYPETYPECYCLING, WFS_CDM_TYPERETRACTCASSETTE or WFS_CDM_TYPETYPEJECTCASSETTE) 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_TYPETYPEJECTCASSETTE), 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 from this cash unit which are in the reject bin, and which have not been accessible to a customer. This value may be unreliable, since the typical reason for dumping items to the reject cash unit is a suspected pick failure. For reject and retract cash units (usType is WFS_CDM_TYPETYPEJECTCASSETTE 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 may trigger threshold events. 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 may trigger threshold events. This value is persistent.

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

usStatus
Supplies the status of the cash unit as one of the following values:
CWA 16374-5:2011 (E)

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_TYPERETRACTCASSETTE.
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_TYPERETRACTCASSETTE.
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_TYPERETRACTCASSETTE.
WFS_CDM_STATCUEMPTY | The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations. This value does not apply to cash units where usType is WFS_CDM_TYPEREJECTCASSETTE or WFS_CDM_TYPERETRACTCASSETTE.
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. 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_STATCUOK</td>
<td>The cash unit is in a good state.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUFULL</td>
<td>The cash unit is full. This value only applies to cash units where usType is WFS_CDM_TYPEREJECT-CASSETTE or WFS_CDM_TYPERETRACT-CASSETTE.</td>
</tr>
</tbody>
</table>
WFS_CDM_STATCUHIGH  The cash unit is almost full (reached or exceeded threshold defined by \textit{ulMaximum}). This value only applies to cash units where \textit{usType} is WFS_CDM_TYPEREJECTCASSETTE or WFS_CDM_TYPERETRACTCASSETTE.

WFS_CDM_STATCULOW   The cash unit is almost empty. This value does not apply to cash units where \textit{usType} is WFS_CDM_TYPEREJECTCASSETTE or WFS_CDM_TYPERETRACTCASSETTE.

WFS_CDM_STATCUEMPTY The cash unit is empty, or insufficient items in the cash unit are preventing further dispense operations. This value does not apply to cash units where \textit{usType} is WFS_CDM_TYPEREJECTCASSETTE or WFS_CDM_TYPERETRACTCASSETTE.

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.

\textit{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.

\textit{ulDispensedCount} 
As defined by the logical \textit{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.

\textit{ulPresentedCount} 
As defined by the logical \textit{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.

\textit{ulRetractedCount} 
As defined by the logical \textit{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.

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

\textbf{Comments}  None.
4.4 WFS-INF_CDM_TELLER_INFO

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

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

Input Param
LPWFSCDMTELLERINFO lpTellerInfo;

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

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

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

This 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_POSNULL</td>
<td>No position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSLEFT</td>
<td>Left position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSRIGHT</td>
<td>Right position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSCENTER</td>
<td>Center position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSTOP</td>
<td>Top position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSBOTTOM</td>
<td>Bottom position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSFRONT</td>
<td>Front position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSREAR</td>
<td>Rear position is assigned to the teller.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_POSNULL</td>
<td>No position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSLEFT</td>
<td>Left position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSRIGHT</td>
<td>Right position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSCENTER</td>
<td>Center position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSTOP</td>
<td>Top position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSBOTTOM</td>
<td>Bottom position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CDM_POSFRONT</td>
<td>Front position is assigned to the teller.</td>
</tr>
</tbody>
</table>
WFS_CDM_POSREAR

Rear position is assigned to the teller.

**lppTellerTotals**

Pointer to a NULL-terminated array of pointers to WFSCDMTELLERTOTALS structures.

```c
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_INVALIDCURRENCY</td>
<td>Specified currency not currently available.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDTELLERID</td>
<td>Invalid teller ID.</td>
</tr>
</tbody>
</table>

**Comments**

None.
4.5 **WFS_INF_CDM_CURRENCY_EXP**

**Description**
This command returns each exponent assigned to each currency known to the Service Provider.

**Input Param**
None.

**Output Param**
LPWFSFSCDMCURRENCYEXP *lppCurrencyExp;

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

```c
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:

\[ \text{<cash\_amount>} = \text{<money\_amount\_parameter>} \times 10^{\text{sExponent}} \]

**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 \times 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.
4.6 WFS_INF_CDM_MIX_TYPES

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

Input Param
None.

Output Param
LPWFSCDMMIXTYPE *lppMixTypes;
Pointer to a NULL-terminated array of pointers to 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_MIXALGORITHM</td>
<td>Mix algorithm.</td>
</tr>
<tr>
<td>WFS_CDM_MIXTABLE</td>
<td>Mix table.</td>
</tr>
</tbody>
</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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_MIX_MINIMUM_NUMBER_OF_BILLS</td>
<td>Select a mix requiring the minimum possible number of items.</td>
</tr>
<tr>
<td>WFS_CDM_MIX_EQUAL_EMPTYING_OF_CASH_UNITS</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_CDM_MIX_MAXIMUM_NUMBER_OF_CASH_UNITS</td>
<td>The denomination will be selected based upon criteria which ensures the maximum number of different logical cash units are used.</td>
</tr>
</tbody>
</table>

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

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

Comments
None.
4.7 WFS_INF_CDM_MIX_TABLE

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

Input Param
LPUSHORT lpusMixNumber;

lpusMixNumber
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_INVALIDMIXNUMBER</td>
<td>The lpusMixNumber parameter does not correspond to a defined mix table.</td>
</tr>
</tbody>
</table>

Comments
None.
4.8 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. 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 output position the items were presented or dispensed to as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_POSNULL</td>
<td>The items were presented according to the default configuration.</td>
</tr>
<tr>
<td>WFS_CDM_POSLEFT</td>
<td>The items were presented to the left output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSRIGHT</td>
<td>The items were presented to the right output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSCENTER</td>
<td>The items were presented to the center output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSTOP</td>
<td>The items were presented to the top output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSBOTTOM</td>
<td>The items were presented to the bottom output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSFRONT</td>
<td>The items were presented to the front output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSREAR</td>
<td>The items were presented to the rear output position.</td>
</tr>
</tbody>
</table>

**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 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 mixed currencies were dispensed the uAmount 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_PRESENTED</td>
<td>The items were presented. This status is set as soon as the customer has access to the items.</td>
</tr>
<tr>
<td>WFS_CDM_NOTPRESENTED</td>
<td>The customer has not had access to the items.</td>
</tr>
<tr>
<td>WFS_CDM_UNKNOWN</td>
<td>It is not known if the customer had access to the items.</td>
</tr>
</tbody>
</table>
**lpszExtra**
Pointer to a list of vendor-specific, or any other extended, information. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_UNSUPPOSITION</td>
<td>The specified output position is not supported.</td>
</tr>
</tbody>
</table>

**Comments**
None.
5. Execute Commands

5.1 WFS_CMD_CDM_DENOMINATE

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

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

If items of differing currencies are to be included in the same denomination then the currency field must be an array of three ASCII 0x20h characters, the amount must be 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 WFSCDCMCAPS 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**

LPWFSCDCMDENOMINATE lpDenominate;

typedef struct _wfs_cdm_denominate
{
    USHORT    usTellerID;
    USHORT    usMixNumber;
    LPWFSCDCMDENOMINATION  lpDenomination;
} WFSCDCMDENOMINATE, *LPWFSCDCMDENOMINATE;

*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 WFSCDCMDENOMINATION structure, describing the contents of the denomination operation.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_INVALIDCURRENCY</td>
<td>There are no cash units in the CDM of the currency specified in the cCurrencyID field of the input parameter.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDTELLERID</td>
<td>Invalid teller ID. This error will never be generated by a Self-Service CDM.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>There is a problem with a cash unit. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDDENOMINATION</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDMIXNUMBER</td>
<td>Unknown mix algorithm.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOCURRENCYMIX</td>
<td>The cash units specified in the denomination were not all of the same currency.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOTDISPENSABLE</td>
<td>The amount is not dispensable by the CDM.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_TOO_MANY_ITEMS</td>
<td>The request requires too many items to be dispensed.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state (see section WFS_CMD_CDM_START_EXCHANGE).</td>
</tr>
</tbody>
</table>
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_EXEE_CDM_CASHUNITERROR</td>
<td>An error occurred while attempting to denominate from the cash unit specified by the event.</td>
</tr>
</tbody>
</table>

Comments  
None.
5.2 WFS_CMD_CDM_DISPENSE

Description
This command performs the dispensing of items to the customer. The command provides the same functionality as the WFS_CMD_CDM_DENOMINATE command plus the additional functionality of dispensing the items. If items of differing currencies are to be dispensed then the currency field must be an array of three ASCII 0x20h characters, the amount must be 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.

Input Param

LPWFSCMDDISPENSE lpDispense;

typedef struct _wfs_cdm_dispense
{
    USHORT    usTellerID;
    USHORT    usMixNumber;
    WORD     fwPosition;
    BOOL     bPresent;
    LPWFSCMDDENOMINATION  lpDenomination;
} WFSCMDDISPENSE, *LPWFSCMDDISPENSE;

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:

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

**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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_INVALIDCURRENCY</td>
<td>There are no cash units in the CDM of the currency specified in the <em>cCurrencyID</em> field of the input parameter.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDTELLERID</td>
<td>Invalid teller ID. This error will never be generated by a Self-Service CDM.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>There is a problem with a cash unit. A WFS_EXEE_CDM_CASHUNITERROR execute event is posted with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDDENOMINATION</td>
<td>The sum of the values for cash box and cash units was greater than the amount specified.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDMIXNUMBER</td>
<td>Mix algorithm is not known.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOCURRENCYMIX</td>
<td>Cash units containing two or more different currencies were selected.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOTDISPENSABLE</td>
<td>The amount is not dispensable by the CDM.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_TOOMANYITEMS</td>
<td>The request would require too many items to be dispensed. This error is also generated if bPresent is FALSE and sub-dispensing is required.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_UNSUPPOSITION</td>
<td>The specified output position is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_SAFEDOOROPEN</td>
<td>The safe door is open. This device requires the safe door to be closed in order to perform this operation.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
</tbody>
</table>
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 hPresent field of the WFSCDMDispense structure is TRUE, the following error codes can also be returned:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_SHUTTERNOTOPEN</td>
<td>The shutter is not open or did not open when it should have. No items presented.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_PRERRORNOITEMS</td>
<td>An error occurred while items were being moved to the exit slot - no items are presented.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_PRERRORITEMS</td>
<td>An error occurred while items were being moved to the exit slot - at least some of the items have been presented.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_PRERRORUNKNOWN</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CDM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_DELAYEDDISPENSE</td>
<td>The dispense operation will be delayed by the specified time.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_STARTDISPENSE</td>
<td>Fired when the delayed dispense operation starts.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_CASHUNITERROR</td>
<td>A cash unit caused an error during a dispense operation.</td>
</tr>
<tr>
<td>WFS_SRVE_CDM_ITEMSTAKEN</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_PARTIALDISPENSE</td>
<td>Indicates that the dispense operation is to be divided into several sub-dispense operations.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_SUBDISPENSEOK</td>
<td>A sub-dispense operation was completed successfully.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_INCOMPLETEDISPFENSE</td>
<td>It has not been possible to dispense the entire denomination but part of the denomination has been dispensed, whether on the intermediate stacker or in customer access. The return error code will be WFS_ERR_CDM_NOTDISPENSABLE.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_NOTEERROR</td>
<td>An item detection error has occurred.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_INPUT_P6</td>
<td>ECB6 level 2 and/or level 3 notes have been detected.</td>
</tr>
</tbody>
</table>

Comments  None.
5.3 WFS_CMD_CDM_COUNT

**Description**

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

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

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

**Input Param**

`LPWFSCDMPHYSICALCU lpPhysicalCU;`

```c
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_POSNULL</td>
<td>Output location is determined by Service Provider.</td>
</tr>
<tr>
<td>WFS_CDM_POSLEFT</td>
<td>Present items to left side of device.</td>
</tr>
<tr>
<td>WFS_CDM_POSRIGHT</td>
<td>Present items to right side of device.</td>
</tr>
<tr>
<td>WFS_CDM_POSCENTER</td>
<td>Present items to center output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSTOP</td>
<td>Present items to the top output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSBOTTOM</td>
<td>Present items to the bottom output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSFRONT</td>
<td>Present items to the front output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSREAR</td>
<td>Present items to the rear output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSTOP</td>
<td>Reject bin is used as output location.</td>
</tr>
</tbody>
</table>

`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;`

```c
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:

```c
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_STATCUOK</td>
<td>The cash unit is in a good state.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUFULL</td>
<td>The cash unit is full.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUHIGH</td>
<td>The cash unit is almost full (reached or exceeded the threshold defined by</td>
</tr>
<tr>
<td></td>
<td>WFSCDMCASHUNIT.ulMaximum).</td>
</tr>
<tr>
<td>WFS_CDM_STATCULOW</td>
<td>The cash unit is almost empty.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUEMPTY</td>
<td>The cash unit is empty.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUNOP</td>
<td>The cash unit is inoperative.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUMISSING</td>
<td>The cash unit is missing.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUNOVAL</td>
<td>The values of the specified cash unit are not available.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUNOREF</td>
<td>There is no reference value available for the notes in this cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUMANIP</td>
<td>The cash unit has been inserted (including removal followed by a</td>
</tr>
<tr>
<td></td>
<td>reinsertion) when the device was not in the exchange state. This cash</td>
</tr>
<tr>
<td></td>
<td>unit cannot be dispensed from.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>A cash unit caused a problem. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_SAFEDOOROPEN</td>
<td>The safe door is open. This device requires the safe door to be closed in order to perform this operation.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_EXEE_CDM_CASHUNITERROR</td>
<td>A cash unit caused an error during the count operation.</td>
</tr>
<tr>
<td>WFS_SRVE_CDM_ITEMSTAKEN</td>
<td>The items emptied to the output location have been removed by the user.</td>
</tr>
<tr>
<td>WFS_SRVE_CDM_ITEMSPRESENTED</td>
<td>Items have been emptied to the output location. These items may need to be removed from the output location before the operation can continue.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_NOTEERROR</td>
<td>An item detection error has occurred.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_INPUT_P6</td>
<td>ECB6 Level 2 and/or level 3 notes have been detected.</td>
</tr>
</tbody>
</table>
Comments
None.
5.4 WFS_CMD_CDM_PRESENT

Description
This command will move items to the exit position for removal by the user. If a shutter exists, then it will be implicitly controlled during the present operation, 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. 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:

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_SHUTTERNOTOPEN</td>
<td>The shutter did not open when it should have. No items presented.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_SHUTTEROPEN</td>
<td>The shutter is open when it should be closed. No items presented.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOITEMS</td>
<td>There are no items on the stacker.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_PRERRORNOITEMS</td>
<td>There was an error during the present operation - no items were presented.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_PRERRORITEMS</td>
<td>There was an error during the present operation - at least some of the items were presented.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_PRERRORUNKNOWN</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CDM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
<tr>
<td>WFS_SRVE_CDM_ITEMSTAKEN</td>
<td>The items have been removed by the user.</td>
</tr>
<tr>
<td></td>
<td>This event is generated after the completion of the present operation.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_INPUT_P6</td>
<td>ECB6 level 2 and/or level 3 notes have been detected.</td>
</tr>
</tbody>
</table>

Comments
None.
5.5 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>A reject cash unit caused a problem. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOITEMS</td>
<td>There were no items on the stacker.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CDM_CASHUNITTHRESHOLD</td>
<td>A reject bin threshold condition has been reached.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_CASHUNITERROR</td>
<td>A cash unit caused an error during the reject operation.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_INPUT_P6</td>
<td>ECB6 level 2 and/or level 3 notes have been detected.</td>
</tr>
</tbody>
</table>

**Comments**
None.
### 5.6 WFS_CMD_CDM_RETRACT

**Description**

This command will retract items which may have been in customer access. Retracted items will be moved to either a retract cash unit, 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.

For cash recycler implementations with a note handling standard it is recommended to use the WFS_CMD_CIM_RETRACT command instead of this command.

**Input Param**

`LPWFSCDMRETRACT lpRetract;`

```c
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_POSNULL</td>
<td>The default configuration information should be used.</td>
</tr>
<tr>
<td>WFS_CDM_POSLEFT</td>
<td>Retract items from the left output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSRIGHT</td>
<td>Retract items from the right output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSCENTER</td>
<td>Retract items from the center output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSTOP</td>
<td>Retract items from the top output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSBOTTOM</td>
<td>Retract items from the bottom output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSFRONT</td>
<td>Retract items from the front output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSREAR</td>
<td>Retract items from the rear output position.</td>
</tr>
</tbody>
</table>

*usRetractArea*

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_RA_RETRACT</td>
<td>Retract the items to a retract cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_RA_TRANSPORT</td>
<td>Retract the items to the transport.</td>
</tr>
<tr>
<td>WFS_CDM_RA_STACKER</td>
<td>Retract the items to the intermediate stacker area.</td>
</tr>
<tr>
<td>WFS_CDM_RA_REJECT</td>
<td>Retract the items to a reject cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_RA_ITEMCASSETTE</td>
<td>Retract the items to the item cassettes, i.e. cassettes that can be dispensed from.</td>
</tr>
</tbody>
</table>
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.

```c
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:

```c
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>A retract cash unit caused a problem. A WFS_ERR_CDM_CASHUNITERROR event will be posted with the details.</td>
</tr>
</tbody>
</table>
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.

**Events**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CDM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in a retract or reject cash unit.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_CASHUNITERROR</td>
<td>An error occurred while attempting to retract to a retract, reject or item cash unit.</td>
</tr>
<tr>
<td>WFS_SRVE_CDM_ITEMSTAKEN</td>
<td>The items presented have been removed by the user.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_INPUT_P6</td>
<td>ECB6 level 2 and/or level 3 notes have been detected.</td>
</tr>
</tbody>
</table>

**Comments**

None.
5.7 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_POSNULL</td>
<td>The default configuration information should be used.</td>
</tr>
<tr>
<td>WFS_CDM_POSLEFT</td>
<td>Open the shutter at the left output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSRIGHT</td>
<td>Open the shutter at the right output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSCENTER</td>
<td>Open the shutter at the center output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSTOP</td>
<td>Open the shutter at the top output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSBOTTOM</td>
<td>Open the shutter at the bottom output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSFRONT</td>
<td>Open the shutter at the front output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSREAR</td>
<td>Open the shutter at the rear output position.</td>
</tr>
</tbody>
</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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_SHUTTERNOTOPEN</td>
<td>The shutter failed to open.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_SHUTTEROPEN</td>
<td>The shutter was already open.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
</tbody>
</table>

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

Comments
None.
5.8 WFS_CMD_CDM_CLOSE_SHUTTER

Description
This command closes the shutter.

Input Param
LPWORD lpfwPosition;

*lpfwPosition*
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_POSNULL</td>
<td>The default configuration information should be used.</td>
</tr>
<tr>
<td>WFS_CDM_POSLEFT</td>
<td>Close the shutter at the left output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSRIGHT</td>
<td>Close the shutter at the right output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSCENTER</td>
<td>Close the shutter at the center output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSTOP</td>
<td>Close the shutter at the top output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSBOTTOM</td>
<td>Close the shutter at the bottom output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSFRONT</td>
<td>Close the shutter at the front output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSREAR</td>
<td>Close the shutter at the rear output position.</td>
</tr>
</tbody>
</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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_SHUTTERCLOSED</td>
<td>The shutter was already closed.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_SHUTTERNOTCLOSED</td>
<td>The shutter failed to close.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
</tbody>
</table>

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

Comments
None.
5.9 WFS_CMD_CDM_SET_TELLER_INFO

**Description**
This command allows the application to set the teller position and 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;
ttypedef struct _wfs_cdm_teller_update
{
    USHORT usAction;
    LPWFSCDMTELLERDETAILS lpTellerDetails;
} WFSCDMTELLERUPDATE, *LPWFSCDMTELLERUPDATE;

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_CREATE_TELLER</td>
<td>A teller is to be added.</td>
</tr>
<tr>
<td>WFS_CDM_MODIFY_TELLER</td>
<td>Information about an existing teller is to be modified.</td>
</tr>
<tr>
<td>WFS_CDM_DELETE_TELLER</td>
<td>A teller is to be removed.</td>
</tr>
</tbody>
</table>

*lпTellerDetails*
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_INVALIDCURRENCY</td>
<td>The specified currency is not currently available.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDTELLERID</td>
<td>The teller ID is invalid. This error will never be generated by a Self-Service CDM.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The target teller is currently in the middle of an exchange operation.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_SRVE_CDM_TELLERINFOCHANGED</td>
<td>Teller information has been created, modified or deleted.</td>
</tr>
</tbody>
</table>

**Comments**
None.
5.10 WFS_CMD_CDM_SET_CASH_UNIT_INFO

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

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

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

The following fields of the WFSCDMCASHUNIT structure may be updated by this command:
- ulInitialCount
- ulCount
- ulRejectCount
- 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.

Output Param
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_INVALIDTELLERID</td>
<td>Invalid teller ID. This error will never be generated by a Self-Service CDM.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDCASHUNIT</td>
<td>Invalid cash unit.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>A problem occurred with a cash unit. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CDM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
</tbody>
</table>
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.
5.11 WFS_CMD_CDM_START_EXCHANGE

Description

This command puts the CDM in an exchange state, i.e. a state in which cash units can be emptied, replenished, removed or replaced. Other than the updates which can be made via the WFS_CMD_CDM_SET_CASH_UNIT_INFO command 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 IpNoteNumberList 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_EXBYHAND</td>
<td>The cash units will be replenished manually either by filling or emptying</td>
</tr>
<tr>
<td></td>
<td>the cash unit by hand or by replacing the cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_EXTOCASSETTES</td>
<td>Items will be moved from the replenishment container to the bill cash</td>
</tr>
<tr>
<td></td>
<td>units.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_INVALIDTELLERID</td>
<td>Invalid teller ID. This error will never be generated by a Self-Service</td>
</tr>
<tr>
<td></td>
<td>CDM.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>An error occurred with a cash unit while performing the exchange</td>
</tr>
<tr>
<td></td>
<td>operation. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the</td>
</tr>
<tr>
<td></td>
<td>details.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is already in an exchange state.</td>
</tr>
</tbody>
</table>

**Events**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_EXEE_CDM_CASHUNITERROR</td>
<td>A cash unit caused an error.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_NOTEERROR</td>
<td>An item detection error has occurred.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_INPUT_P6</td>
<td>ECB6 level 2 and/or level 3 notes have been detected.</td>
</tr>
</tbody>
</table>

**Comments**

None.
5.12 WFS_CMD_CDM_END_EXCHANGE

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

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**
LPWFS_CDMCUINFO 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>A problem occurred with a cash unit A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOEXCHANGEACTIVE</td>
<td>There is no exchange active.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_EXEE_CDM_CASHUNITERROR</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_SRVE_CDM_CASHUNITINFOCHANGED</td>
<td>A cash unit was changed.</td>
</tr>
<tr>
<td>WFS_USRE_CDM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_NOTEERROR</td>
<td>An item detection error has occurred.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_INPUT_P6</td>
<td>ECB6 level 2 and/or level 3 notes have been detected.</td>
</tr>
</tbody>
</table>

**Comments**

None.
5.13 WFS_CMD_CDM_OPEN_SAFE_DOOR

Description: This command unlocks the safe door or starts the time delay countdown 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
</tbody>
</table>

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

Comments: None.
5.14 WFS_CMD_CDM_CALIBRATE_CASH_UNIT

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

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

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

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>A cash unit caused an error. A</td>
</tr>
<tr>
<td></td>
<td>WFS_EXEE_CDM_CASHUNITERROR event will be sent with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_UNSUPPOSITION</td>
<td>The position specified is not valid.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDCASHUNIT</td>
<td>The cash unit number specified is not valid.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CDM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
</tbody>
</table>
CWA 16374-5:2011 (E)

<table>
<thead>
<tr>
<th>Code</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_SRVE_CDM_CASHUNITINFOCHANGED</td>
<td>A cash unit was changed.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_CASHUNITERROR</td>
<td>A cash unit caused an error.</td>
</tr>
<tr>
<td>WFS_SRVE_CDM_ITEMSTAKEN</td>
<td>The items were removed.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_NOTEERROR</td>
<td>An item detection error has occurred.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_INPUT_P6</td>
<td>ECB6 level 2 and/or level 3 notes have been detected.</td>
</tr>
</tbody>
</table>

**Comments**  None.
5.15 WFS_CMD_CDM_SET_MIX_TABLE

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

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

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_INVALIDMIXNUMBER</td>
<td>The supplied USMixNumber is reserved for a predefined mix algorithm.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDMIXTABLE</td>
<td>The contents of at least one of the defined rows of the mix table is incorrect.</td>
</tr>
</tbody>
</table>

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

Comments
None.
5.16 WFS_CMD_CDM_RESET

Description
This command is used by the application to perform a hardware reset which will attempt to return the CDM device to a known good state. This command does not over-ride a lock obtained 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 WFSCDMCASUNIT.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 WFSCDMCASUNIT.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
In the case of a single cash unit destination this value specifies the cash unit to be used for the storage of any items found, i.e. when items are to be moved to a reject or retract cash unit. In all other cases this value must be zero, i.e. when items are to be moved to item cassettes, the transport, the stacker or an output position.

lpRetractArea
This field is used if items are to be moved to the stacker, the transport, a retract cash unit or to item cassettes. If items are to be moved to a reject cash unit or to an output position then this field must be NULL.

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

fwOutputPosition
This value will be ignored.

usRetractArea
This value specifies the area to which the items are to be moved to. Possible values are:
Value | Meaning
---|---
WFS_CDM_RA_RETRACT | Items will be moved to a retract cash unit. In the case where several cash units of type WFS_CDM_TYPERETRACTCASSETTE exist the usNumber field will define which retract unit the items will be moved to.
WFS_CDM_RA_TRANSPORT | Items will be moved to the transport.
WFS_CDM_RA_STACKER | Items will be moved to the intermediate stacker area.
WFS_CDM_RA_ITEMCASSETTE | Items will be moved 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 WFSCDMCASHINFO. The maximum value of `usIndex` is the sum of the WFSCDMCASHUNIT.ulMaximum of each retract cash unit. If `usRetractArea` is not set to WFS_CDM_RA_RETRACT the value of this field is ignored.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_POSNULL</td>
<td>The default configuration.</td>
</tr>
<tr>
<td>WFS_CDM_POSLEFT</td>
<td>The left output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSRIGHT</td>
<td>The right output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSCENTER</td>
<td>The center output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSTOP</td>
<td>The top output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSBOTTOM</td>
<td>The bottom output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSFRONT</td>
<td>The front output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSREAR</td>
<td>The rear output position.</td>
</tr>
</tbody>
</table>

**Output Param**
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>A cash unit caused an error. One or more WFS_EXEE_CDM_CASHUNITERROR events will be sent with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDCASHUNIT</td>
<td>The cash unit number specified is not valid.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CDM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_CASHUNITERROR</td>
<td>A cash unit caused an error.</td>
</tr>
<tr>
<td>WFS_SRVE_CDM_MEDIADECTECTED</td>
<td>Media has been found in the device.</td>
</tr>
<tr>
<td>WFS_SRVE_CDM_ITEMSTAKEN</td>
<td>The items presented have been removed by the user.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_INPUT_P6</td>
<td>ECB6 level 2 and/or level 3 notes have been detected.</td>
</tr>
</tbody>
</table>

**Comments**
None.
5.17 WFS_CMD_CDM_TEST_CASH_UNITS

Description
This command is used to test cash units following replenishment. All physical cash units 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_ERR_CDM_CASHUNITERROR event will be sent for any logical cash unit which has one or more physical cash units which can not 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_ERR_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_ERR_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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_SHUTTERNOTOPEN</td>
<td>The shutter is not open or did not open when it should have. No items presented.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_SHUTTEROPEN</td>
<td>The shutter is open when it should be closed. No items presented.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDCASHUNIT</td>
<td>The cash unit number specified is not valid.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_PRERRORNOITEMS</td>
<td>There was an error during the present operation - no items were presented.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_PRERRORITEMS</td>
<td>There was an error during the present operation - at least some of the items were presented.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_PRERRORUNKNOWN</td>
<td>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.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CDM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
</tbody>
</table>
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: ECB6 level 2 and/or level 3 notes have been detected.

Comments: None.
### 5.18 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 and the color. When an application tries to use a color that is not supported then the Service Provider will return the generic error WFS_ERR_UNSUPP_DATA.

**Input Param**
LPWFSFSCDMSETGUIDLIGHT lpSetGuidLight;

typedef struct _wfs_cdm_set_guidlight
{
    WORD wGuidLight;
    DWORD dwCommand;
} WFSFSCDMSETGUIDLIGHT, *LPWFSFSCDMSETGUIDLIGHT;

- **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, and optionally one type C. If no value of type C is specified then the default color is used. The Service Provider determines which color is used as the default color.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_GUIDANCE_OFF</td>
<td>The light indicator is turned off.</td>
<td>A</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_SLOW_FLASH</td>
<td>The light indicator is set to flash slowly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_MEDIUM_FLASH</td>
<td>The light indicator is set to flash medium frequency.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_QUICK_FLASH</td>
<td>The light indicator is set to flash quickly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_CONTINUOUS</td>
<td>The light indicator is turned on continuously (steady).</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_RED</td>
<td>The light indicator color is set to red.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_GREEN</td>
<td>The light indicator color is set to green.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_YELLOW</td>
<td>The light indicator color is set to yellow.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_BLUE</td>
<td>The light indicator color is set to blue.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_CYAN</td>
<td>The light indicator color is set to cyan.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_MAGENTA</td>
<td>The light indicator color is set to magenta.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CDM_GUIDANCE_WHITE</td>
<td>The light indicator color is set to white.</td>
<td>C</td>
</tr>
</tbody>
</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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_INVALID_PORT</td>
<td>An attempt to set a guidance light to a new value was invalid because the guidance light does not exist.</td>
</tr>
</tbody>
</table>

**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.
5.19 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;
} WFSPOWERSAVECONTROL, *LPWFSPOWERSAVECONTROL;

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

Output Param
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_POWERSAVETOOSHORT</td>
<td>The power saving mode has not been activated because the device is not able to resume from the power saving mode within the specified usMaxPowerSaveRecoveryTime value.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_POWERSAVEMEDIAPRESENT</td>
<td>The power saving mode has not been activated because media is present inside the device.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
</tbody>
</table>

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

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

Comments
None.
5.20 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 command, 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* state, then this execution will have no effect and will complete with WFS_SUCCESS.

**Input Param**

```c
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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_START</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_CDM_STOP</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Output Param**
None.

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

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

**Comments**
None.
6. Events

6.1 WFS_SRVE_CDM_SAFEDOOROPEN

<table>
<thead>
<tr>
<th>Description</th>
<th>This service event is generated when the safe door has been opened.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Param</td>
<td>None.</td>
</tr>
<tr>
<td>Comments</td>
<td>None.</td>
</tr>
</tbody>
</table>
### 6.2 WFS_SRVE_CDM_SAFEDOORCLOSED

<table>
<thead>
<tr>
<th>Description</th>
<th>This service event is generated when the safe door has been closed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Param</td>
<td>None.</td>
</tr>
<tr>
<td>Comments</td>
<td>None.</td>
</tr>
</tbody>
</table>
6.3 **WFS_USRE_CDM_CASHUNITTHRESHOLD**

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

**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.
6.4 WFS_SRVE_CDM_CASHUNITINFOCHANGED

Description
This service event is generated when information about a physical or logical cash unit has changed. For instance, a physical cash unit may have been removed or inserted. This event will also be posted 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), or if the WFS_USRE_CDM_CASHUNITTHRESHOLD is more appropriate:

- 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.
6.5 WFS_SRVE_CDM_TELLERINFOCHANGED

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

**Event Param**
LPUSHORT lpusTellerID;

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

**Comments**
None.
### 6.6 WFS_EXEE_CDM_DELAYEDDISPENSE

<table>
<thead>
<tr>
<th>Description</th>
<th>This execute event is generated if the start of a dispense operation has been delayed.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Event Param</strong></td>
<td>LPULONG lpulDelay;</td>
</tr>
<tr>
<td><em>lpulDelay</em></td>
<td>Pointer to an unsigned long holding the time in milliseconds by which the dispense operation will be delayed.</td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td>None.</td>
</tr>
</tbody>
</table>

6.7  WFS_EXEE_CDM_STARTDISPENSE

<table>
<thead>
<tr>
<th>Description</th>
<th>This execute event is generated when a delayed dispense operation begins.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Param</td>
<td>LPREQUESTID lpReqID;</td>
</tr>
<tr>
<td></td>
<td>\hspace{1cm} \textit{lpReqID}</td>
</tr>
<tr>
<td></td>
<td>Pointer to the \textit{RequestID} of the original dispense command.</td>
</tr>
<tr>
<td>Comments</td>
<td>None.</td>
</tr>
</tbody>
</table>
6.8 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**
```
LPWFS_CDMCUERROR lpCashUnitError;
```

typedef struct _wfs_cdm_cu_error
{
    WORD     wFailure;
    LPWFS_CDMCASHUNIT lpCashUnit;
} WFS_CDMCUERROR, *LPWFS_CDMCUERROR;

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_CASHUNITEMPTY</td>
<td>Specified cash unit is empty.</td>
</tr>
<tr>
<td>WFS_CDM_CASHUNITERROR</td>
<td>Specified cash unit has malfunctioned.</td>
</tr>
<tr>
<td>WFS_CDM_CASHUNITFULL</td>
<td>Specified cash unit is full.</td>
</tr>
<tr>
<td>WFS_CDM_CASHUNITLOCKED</td>
<td>Specified cash unit is locked.</td>
</tr>
<tr>
<td>WFS_CDM_CASHUNITINVALID</td>
<td>Specified cash unit is invalid.</td>
</tr>
<tr>
<td>WFS_CDM_CASHUNITNOTCONF</td>
<td>An attempt has been made to change the settings of a self-configuring cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_CASHUNITINVALID</td>
<td>Specified cash unit is not configured.</td>
</tr>
</tbody>
</table>

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

**Comments**
None.
6.9  WFS_SRVE_CDM_ITEMSTAKEN

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_POSNULL</td>
<td>The default configuration.</td>
</tr>
<tr>
<td>WFS_CDM_POSLEFT</td>
<td>The left output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSRIGHT</td>
<td>The right output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSCENTER</td>
<td>The center output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSSTOP</td>
<td>The top output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSBOTTOM</td>
<td>The bottom output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSFRONT</td>
<td>The front output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSREAR</td>
<td>The rear output position.</td>
</tr>
</tbody>
</table>

Comments  None.
6.10 WFS_SRVE_CDM_COUNTS_CHANGED

**Description**
This service event is generated if the device is a compound device together with a CIM and the counts in a shared cash unit have changed as a result of any CIM operation other than WFS_CMD_CIM_SET_CASH_UNIT_INFO and WFS_CMD_CIM_END_EXCHANGE.

**Event Param**

```
LPWFSFSCDMCOUNTSCHANGED lpCountsChanged;
```

typedef struct _wfs_cdm_counts_changed
{
    USHORT usCount;
    LPUCHAR lpusCUNumList;
} WFSFSCDMCOUNTSCHANGED, *LPWFSFSCDMCOUNTSCHANGED;

*usCount*
The size of *lpusCUNumList*.

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

**Comments**
None.
6.11 WFS_EXEE_CDM_PARTIALDISPENSE

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

Event Param
LPUSHORT lpusDispNum;

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

Comments
None.
6.12 WFS_EXEE_CDM_SUBDISPENSEOK

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

**Event Param**
LPWFSCDMDENOMINATION lpDenomination;

*lpDenomination*

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

**Comments**
None.
6.13 WFS_EXEE_CDM_INCOMPLETEDISPENSE

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

**Event Param**  LPWFSCDMDENOMINATION lpDenomination;

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

**Comments**  None.
### 6.14 WFS_EXEE_CDM_NOTEERROR

**Description**
This execute event specifies the reason for a 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_DOUBLENOTEDETECTED</td>
<td>Double notes have been detected.</td>
</tr>
<tr>
<td>WFS_CDM_LONGNOTEDETECTED</td>
<td>A long note has been detected.</td>
</tr>
<tr>
<td>WFS_CDM_SKEWEDNOTE</td>
<td>A skewed note has been detected.</td>
</tr>
<tr>
<td>WFS_CDM_INCORRECTCOUNT</td>
<td>An item counting error has occurred.</td>
</tr>
<tr>
<td>WFS_CDM_NOTESTOOCLOSE</td>
<td>Notes have been detected as being too close.</td>
</tr>
<tr>
<td>WFS_CDM_OTHERNOTEERROR</td>
<td>An item error not covered by the other values has been detected.</td>
</tr>
<tr>
<td>WFS_CDM_SHORTNOTEDETECTED</td>
<td>Short notes have been detected.</td>
</tr>
</tbody>
</table>

**Comments**
None.
### 6.15 WFS_SRVE_CDM_ITEMSPRESENTED

<table>
<thead>
<tr>
<th><strong>Description</strong></th>
<th>This service event specifies that items have been presented to the user during a count operation and need to be taken.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Event Param</strong></td>
<td>None.</td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td>None.</td>
</tr>
</tbody>
</table>
### 6.16 WFS_SRVE_CDM_MEDIADETECTED

**Description**
This service event is generated if media is detected during a reset (WFS_CMD_CDM_RESET). The parameter on the event informs the application of the position of the media after the reset completes. If the device has been unable to successfully move the items found then this parameter will be NULL.

**Event Param**
LPWFSCDMITEMPOSITION *lpItemPosition;
For a description of this parameter see section WFS_CMD_CDM_RESET.

**Comments**
None.
### 6.17 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. Details about the notes detected and their associated signatures are obtained through the CIM interface.

**Event Param**
None.

**Comments**
None.
6.18 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:

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

Comments
None.
6.19 WFS_SRVE_CDM_POWER_SAVE_CHANGE

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

**Event Param**

```c
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. 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:

<table>
<thead>
<tr>
<th>Step</th>
<th>End-User</th>
<th>Application</th>
<th>XFS SP</th>
<th>CDM Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>User wants to dispense 40 USD.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td>WFS_CMD_CDM_DISPENSE command issued.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td></td>
<td>Determines that a single “hardware level” dispense command can be issued for full dispense request.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td></td>
<td>“Hardware level” dispense command issued.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td></td>
<td>Items presented.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td>WFS_CMD_CDM_DISPENSE completes successfully.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td>User takes bills.</td>
<td></td>
<td>WFS_SRVE_CDM_ITEMSTAKEN event generated.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td></td>
<td>WFS_SRVE_CDM_ITEMSTAKEN event generated.</td>
<td></td>
</tr>
</tbody>
</table>
Sub-Dispensing Is Required - Command Successful
This table illustrates a successful WFS_CMD_CDM_DISPENSE command where sub-dispensing is required:

<table>
<thead>
<tr>
<th>Step</th>
<th>End-User</th>
<th>Application</th>
<th>XFS SP</th>
<th>CDM Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>User wants to dispense 130 USD in 1 USD denominations.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td>WFS_CMD_CDM_DISPENSE command issued.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td>Three “hardware level” dispense commands are required. CDM hardware is limited to dispensing 50 bills in any single “hardware level” dispense.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td>WFS_EXEE_CDM_PARTIAL-DISPENSE event generated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td>“Hardware level” dispense command issued for 50 USD.</td>
<td></td>
<td>Items presented.</td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td>WFS_SRVE_CDM_SUBDISPENSE-OK event generated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td>User takes bills.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td>WFS_SRVE_CDM_ITEMSTAKEN event generated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td>“Hardware level” dispense command issued for 50 USD.</td>
<td></td>
<td>Items presented.</td>
</tr>
<tr>
<td>10.</td>
<td></td>
<td>WFS_SRVE_CDM_SUBDISPENSE-OK event generated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td></td>
<td>User takes bills.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td></td>
<td>WFS_SRVE_CDM_ITEMSTAKEN event generated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td></td>
<td>“Hardware level” dispense command issued for 30 USD.</td>
<td></td>
<td>Items presented.</td>
</tr>
<tr>
<td>14.</td>
<td></td>
<td>WFS_SRVE_CDM_SUBDISPENSE-OK event generated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td></td>
<td>WFS_CMD_CDM_DISPENSE completes successfully.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td></td>
<td>User takes bills.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.</td>
<td></td>
<td>WFS_SRVE_CDM_ITEMSTAKEN event generated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>18.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 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.

<table>
<thead>
<tr>
<th>Step</th>
<th>End-User</th>
<th>Application</th>
<th>XFS SP</th>
<th>CDM Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>User wants to dispense 130 USD in 1 USD denominations.</td>
<td>WFS_CMD_CDM_DISPENSE command issued.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td>Three “hardware level” dispense commands are required. CDM hardware is limited to dispensing 50 bills in any single “hardware level” dispense command.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td>WFS_EXEE_CDM_PARTIAL-DISPENSE event generated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td>“Hardware level” dispense command issued for 50 USD.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td>Items presented</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td>WFS_SRVE_CDM_SUBDISPENSE-OK event generated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>User takes bills.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td>WFS_SRVE_CDM_ITEMSTAKEN event generated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td></td>
<td>“Hardware level” dispense command issued for 50 USD.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td></td>
<td>Items presented.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td></td>
<td>WFS_SRVE_CDM_SUBDISPENSE-OK event generated.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>User does not take bills.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td></td>
<td>Timeout occurs waiting on end-user to take bills.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td></td>
<td>WFS_CMD_CDM_DISPENSE completes with WFS_ERR_CDM_ITEMSTAKEN</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8. Rules for Cash Unit Exchange

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

If a cash unit is removed from the CDM outside of the Start/End Exchange operations 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.
#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             0x1403 /* Version 3.20 */
#define WFS_SERVICE_CLASS_NAME_CDM                "CDM"
#define CDM_SERVICE_OFFSET                        (WFS_SERVICE_CLASS_CDM * 100)

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

/* CDM Execute Commands */
#define WFS_CMD_CDM_DENOMINATE                    (CDM_SERVICE_OFFSET + 1)
#define WFS_CMD_CDM_DISPENSE                      (CDM_SERVICE_OFFSET + 2)
#define WFS_CMD_CDM_PRESENT                       (CDM_SERVICE_OFFSET + 3)
#define WFS_CMD_CDM_REJECT                        (CDM_SERVICE_OFFSET + 4)
#define WFS_CMD_CDM_RETRACT                       (CDM_SERVICE_OFFSET + 5)
#define WFS_CMD_CDM_OPEN_SHUTTER                  (CDM_SERVICE_OFFSET + 7)
#define WFS_CMD_CDM_CLOSE_SHUTTER                 (CDM_SERVICE_OFFSET + 8)
#define WFS_CMD_CDM_SET_TELLER_INFO               (CDM_SERVICE_OFFSET + 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_CASH_UNIT_INFO            (CDM_SERVICE_OFFSET + 23)
#define WFS_CMD_CDM_COUNT                         (CDM_SERVICE_OFFSET + 24)
#define WFS_CMD_CDM_SET_GUIDANCE_LIGHT            (CDM_SERVICE_OFFSET + 25)
#define WFS_CMD_CDM_PREPARE_DISPENSE              (CDM_SERVICE_OFFSET + 26)

/* 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_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_EXEE_CDM_CDM_COUNTS_CHANGED (CDM_SERVICE_OFFSET + 14)
#define WFS_SRVE_CDM_ITEMSPRESENTED (CDM_SERVICE_OFFSET + 15)
#define WFS_SRVE_CDM_COUNTS_CHANGED (CDM_SERVICE_OFFSET + 16)
#define WFS_SRVE_CDM_ITEMSPRESENTED (CDM_SERVICE_OFFSET + 17)
#define WFS_SRVE_CDM_COUNTS_CHANGED (CDM_SERVICE_OFFSET + 18)
#define WFS_SRVE_CDM_ITEMSPRESENTED (CDM_SERVICE_OFFSET + 19)
#define WFS_SRVE_CDM_COUNTS_CHANGED (CDM_SERVICE_OFFSET + 20)

/* 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_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)

/* Values of WFSCDMSTATUS.dwGuidLights [...]
WFSCDMCAPS.dwGuidLights [...]
*/
#define WFS_CDM_GUIDANCE_NOT_AVAILABLE (0x0000)

/* values of WFSCDMSTATUS.wDevicePosition
WFSCDMDVICEPOSITION.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 WFSCDMCAPS.fwType */
#define WFS_CDM_TELLERBILL (0)
#define WFS_CDM_SELFSERVICEBILL (1)
#define WFS_CDM_TELLERCOIN (2)
#define WFS_CDM_SELFSERVICECOIN (3)
/* values of WFSCDMCAPS.fwRetractAreas */
/* values of WFSCDMRETRACT.usRetractArea */
#define WFS_CDM_RA_RETRACT (0x0001)
#define WFS_CDM_RA_TRANSPORT (0x0002)
#define WFS_CDM_RA_STACKER (0x0004)
#define WFS_CDM_RA_REJECT (0x0008)
#define WFS_CDM_RA_NOTSUPP (0x0010)
#define WFS_CDM_RA_ITEMCASSETTE (0x0020)
/* values of WFSCDMCAPS.fwRetractTransportActions */
/* values of WFSCDMCAPS.fwRetractStackerActions */
#define WFS_CDM_PRESENT (0x0001)
#define WFS_CDM_RETRACT (0x0002)
#define WFS_CDM_REJECT (0x0004)
#define WFS_CDM_NOTSUPP (0x0008)
#define WFS_CDM_ITEMCASSETTE (0x0010)
/* values of WFSCDMCAPS.fwMoveItems */
#define WFS_CDM_FROMCU (0x0001)
#define WFS_CDM_TOCU (0x0002)
#define WFS_CDM_TOTRANSPORT (0x0004)
/* values of WFSCDMCASHUNIT.usType */
#define WFS_CDM_TYPENA (1)
#define WFS_CDM_TYPEREJECTCASSETTE (2)
#define WFS_CDM_TYPEBILLCASSETTE (3)
#define WFS_CDM_TYPECOINCYLINDER (4)
#define WFS_CDM_TYPECOINDISPENSER (5)
#define WFS_CDM_TYPERETRACTCASSETTE (6)
#define WFS_CDM_TYPECOUPON (7)
#define WFS_CDM_TYPEDOCUMENT (8)
#define WFS_CDM_TYPERETRACTCASSETTE (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_STATCULOW (4)
#define WFS_CDM_STATCUMISSING (5)
#define WFS_CDM_STATCUNOREF (6)
#define WFS_CDM_STATCUMANIP (7)
#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_MIX_INDIVIDUAL (0)
/* values of WFSCDMMIXTYPE.usSubType (predefined mix algorithms) */
#define WFS_CDM_MIX_MINIMUM_NUMBER_OF_BILLS (1)
#define WFS_CDM_MIX_EQUAL_EMPTYING_OF_CASH_UNITS (2)
#define WFS_CDM_MIX_MAXIMUM_NUMBER_OF_CASH_UNITS (3)
/* values of WFSCDMPRESENTSTATUS.wPresentState */
#define WFS_CDM_PRESENTED (1)
#define WFS_CDM_NOTPRESENTED (2)
#define WFS_CDM_UNKNOWN (3)

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

#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 WFSCDSTATUS.wAntiFraudModule */
#define WFS_CDM_AFMNOTSUPP (0)
#define WFS_CDM_AFMOK (1)
#define WFS_CDM_AFMNOP (2)
#define WFS_CDM_AFMDETECTED (3)
#define WFS_CDM_AFMUNKNOWN (4)

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

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

typedef struct _wfs_cdm_status
{
    WORD     fwDevice;
    WORD     fwSafeDoor;
    WORD     fwDispenser;
    WORD     fwIntermediateStacker;
    LPSTR    lpszExtra;
    DWORD    dwGuidLights[WFS_CDM_GUIDLIGHTS_SIZE];
    WORD     wDevicePosition;
    USHORT   usPowerSaveRecoveryTime;
    WORD     wAntiFraudModule;
} WFSCDMSTATUS, *LPWFSCDMSTATUS;

typedef struct _wfs_cdm_caps
{
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;
}
typedef struct _wfs_cdm_teller_totals
{
    char cCurrencyID[3];
    ULONG ulItemsReceived;
    ULONG ulItemsDispensed;
    ULONG ulCoinsReceived;
    ULONG ulCoinsDispensed;
    ULONG ulCashBoxReceived;
    ULONG ulCashBoxDispensed;
} 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_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;

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

typedef struct _wfs_cdm_cu_error
{
    WORD            wFailure;
    LPWFSCDMCA$HUNIT lpCashUnit;
} WFSCD$MCUERROR, *LPWFSCD$MCUERROR;

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;

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

}         /*extern "C"*/
#endif
#endif  /* __INC_XFSCDM__H */