Extensions for Financial Services (XFS) interface specification - Release 3.0 - Part 19: Cash Dispenser Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

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

© 2000 CEN All rights of exploitation in any form and by any means reserved world-wide for CEN National Members

Ref. No CWA 14050-19:2000 E
# Table of Contents

Foreword

1. General

2. New Chapters
   2.1. References
   2.2. ATM Cash In Transaction Flow – Application Guidelines
   2.3. Rules for Cash Unit Exchange

3. CDM
   3.1. Info Commands moved to the CIM specification
   3.2. New Info Commands
   3.3. Info Commands removed
   3.4. Changes to Existing Info Commands
      3.4.1. WFS_INF_CDM_STATUS
      3.4.2. WFS_INF_CDM_CAPABILITIES
      3.4.3. WFS_INF_CDM_CASH_UNIT_INFO
      3.4.4. WFS_INF_CDM_TELLER_INFO
      3.4.5. WFS_INF_CDM_CURRENCY_EXP
      3.4.6. WFS_INF_CDM_MIX_TYPES
      3.4.7. WFS_INF_CDM_MIX_TABLE
      3.4.8. WFS_INF_CDM_PRESENT_STATUS
   3.5. Execute Commands moved to the CIM specification
      3.5.1. WFS_CMD_CDM_CASH_IN
      3.5.2. WFS_CMD_CDM_CASH_IN_START
      3.5.3. WFS_CMD_CDM_CASH_IN_END
      3.5.4. WFS_CMD_CDM_CASH_IN_ROLLBACK
   3.6. New Execute Commands
      3.6.1. WFS_CMD_CDM_COUNT
      3.6.2. WFS_CMD_CDM_RESET
      3.6.3. WFS_CMD_CDM_TEST_CASH_UNITS
   3.7. Execute Command removed
      3.7.1. WFS_CMD_CDM_CHECK_VANDALISM
      3.7.2. WFS_CMD_CDM_SET_TELLER_POSITIONS
   3.8. Changes to existing Execute Commands
      3.8.1. WFS_CMD_CDM_DENOMINATE
      3.8.2. WFS_CMD_CDM_DISPENSE
      3.8.3. WFS_CMD_CDM_PRESENT
      3.8.4. WFS_CMD_CDM_REJECT
      3.8.5. WFS_CMD_CDM_RETRACT
      3.8.6. WFS_CMD_CDM_OPEN_SHUTTER
      3.8.7. WFS_CMD_CDM_CLOSE_SHUTTER
      3.8.8. WFS_CMD_CDM_SET_TELLER_INFO
      3.8.9. WFS_CMD_CDM_SET_CASH_UNIT_INFO
      3.8.10. WFS_CMD_CDM_START_EXCHANGE
      3.8.11. WFS_CMD_CDM_END_EXCHANGE
      3.8.12. WFS_CMD_CDM_SET_TELLER_POSITIONS
      3.8.13. WFS_CMD_CDM_CHECK_VANDALISM
      3.8.14. WFS_CMD_CDM_SET_TELLER_INFO
      3.8.15. WFS_CMD_CDM_SET_CASH_UNIT_INFO
      3.8.16. WFS_CMD_CDM_START_EXCHANGE
      3.8.17. WFS_CMD_CDM_END_EXCHANGE
3.9. Events moved to the CIM specification ................................................................. 43

3.10. New Events ....................................................................................................... 44

3.11. Events removed ............................................................................................... 45

3.12. Changes to existing Events ............................................................................. 45

3.13. New Sections .................................................................................................. 48

4. CIM ....................................................................................................................... 49

4.1. New Info Commands ...................................................................................... 49

4.2. Changes to Info Commands which previously existed in the CDM ............... 50

4.3. New Execute Commands ............................................................................... 63

4.4. Changes To Execute Commands which previously existed in the CDM ....... 65
4.5. New Events................................................................. 78
4.5.1. WFS_SRVE_CIM_COUNTS_CHANGED.......................... 78
4.5.2. WFS_SRVE_CIM_ITEMSPRESENTED.......................... 79
4.5.3. WFS_SRVE_CIM_ITEMSINSERTED............................ 79
4.5.4. WFS_EXEE_CIM_NOTEERROR................................. 79
4.5.5. WFS_EXEE_CIM_SUBCASHIN................................. 79
4.5.6. WFS_SRVE_CIM_MEDIADETECTED.......................... 80

4.6. Changes to Events which previously existed in the CDM........ 80
4.6.1. WFS_SRVE_CIM_ITEMSTAKEN (former WFS_SRVE_CDM_BILLSTAKEN)....................... 80
4.6.2. WFS_SRVE_CIM_CASHUNITINFOCHANGED.................. 80
4.6.3. WFS_SRVE_CIM_TELLERINFOCHANGED..................... 80
4.6.4. WFS_EXEE_CIM_CASHUNITERROR.......................... 81
4.6.5. WFS_EXEE_CIM_INPUTREFUSE.............................. 81

5. Change to CDM C-Header file........................................ 82
Foreword

This CWA is revision 3.0 of the XFS interface specification.

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

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

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

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

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

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

This CWA was formally approved by the XFS Workshop meeting on 2000-10-18. The specification is continuously reviewed and commented in the CEN/ISSS Workshop on XFS. It is therefore expected that an update of the specification will be published in due time as a CWA, superseding this revision 3.0.

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

Part 1: Application Programming Interface (API) - Service Provider Interface (SPI); Programmer's Reference
Part 2: Service Classes Definition; Programmer's Reference
Part 3: Printer Device Class Interface - Programmer's Reference
Part 4: Identification Card Device Class Interface - Programmer's Reference
Part 5: Cash Dispenser Device Class Interface - Programmer's Reference
Part 6: PIN Keypad Device Class Interface - Programmer's Reference
Part 7: Check Reader/Scanner Device Class Interface - Programmer's Reference
Part 8: Depository Device Class Interface - Programmer's Reference
Part 9: Text Terminal Unit Device Class Interface - Programmer's Reference
Part 10: Sensors and Indicators Unit Device Class Interface - Programmer's Reference
Part 11: Vendor Dependent Mode Device Class Interface - Programmer's Reference
Part 12: Camera Device Class Interface - Programmer's Reference
Part 13: Alarm Device Class Interface - Programmer's Reference
Part 14: Card Embossing Unit Class Interface - Programmer's Reference
Part 15: Cash In Module Device Class Interface - Programmer's Reference
Part 16: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference
Part 17: Printer Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference
Part 18: Identification Card Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference
Part 19: Cash Dispenser Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

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

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

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

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

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

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

In addition to these Programmer's Reference specifications, the reader of this CWA is also referred to a complementary document, called Release Notes. The Release Notes contain clarifications and explanations on the CWA specifications, which are not requiring functional changes. The current version of the Release Notes is available online from http://www.cenorm.be/isss/Workshop/XFS.

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

The 2.00 CDM specification has now been split into two separate specifications: the 3.00 CDM specification which describes cash dispensing functionality and the 3.00 CIM specification which describes cash accepting functionality. Extensive changes have been made to both areas of functionality and are detailed in the CDM and CIM sections below.

Throughout the whole specification the terms “bills”, “coins”, “documents” etc. were replaced by the term “item”. This also effects the names of events (e.g. WFS_SRVE_CDM_BILLSTAKEN is now called WFS_SRVE_CDM_ITEMSTAKEN) and of parameters (e.g. bBillsTakenSensor is now called bItemsTakenSensor).

2. **New Chapters**

2.1. **References**

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer’s Reference Revision 3.0, October 18, 2000

2.2. **ATM Cash In Transaction Flow – Application Guidelines**

2.3. **Rules for Cash Unit Exchange**
3. **CDM**

3.1. **Info Commands moved to the CIM specification**

There were no Info Commands completely moved to the CIM specification.

3.2. **New Info Commands**

There are no new Info Commands.

3.3. **Info Commands removed**

3.3.1. **WFS_INF_CDM_TELLER_POSITIONS**

The information supplied by this command is now included in the WFS_INF_CDM_TELLER_INFO command.

3.4. **Changes to Existing Info Commands**

3.4.1. **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**

```c
LPWFSCDMSTATUS lpStatus;
```

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

**fwDevice**

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

<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>
</tbody>
</table>
WFS_CDM_DEVUSERERROR
The device is present but a person is preventing proper device operation.

WFS_CDM_DEVBUSY
The device is busy and unable to process an execute command at this time.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CDM_DOORNOTSUPPORTED</td>
<td>Physical device has no safe door or 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 but not locked.</td>
</tr>
<tr>
<td>WFS_CDM_DOORLOCKED</td>
<td>Safe door is closed and locked.</td>
</tr>
<tr>
<td>WFS_CDM_DOORUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the door cannot be determined.</td>
</tr>
</tbody>
</table>

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>The dispenser is operational, but one or more of the cash units is in a low, empty or inoperative condition. Items can still be dispensed from at least one of the cash units.</td>
</tr>
<tr>
<td>WFS_CDM_DISPCUSTOP</td>
<td>Due to a cash unit failure dispensing is impossible. The dispenser is operational, but no items can be dispensed because all of the cash units are in an empty or inoperative condition. This state also occurs when a reject/retract cash unit is full or no reject/retract cash unit is present, or an application lock is set on every cash unit.</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 from which items can be dispensed or presented:

```c
typedef struct _wfscdm_position_
{
    ULONG    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>
<tr>
<td>WFS_CDM_POSRIGHT</td>
<td>Right output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSCENTER</td>
<td>Center output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSBILLINPUT</td>
<td>Bill input position.</td>
</tr>
<tr>
<td>WFS_CDM_POSCOINOUTPUT</td>
<td>Coin output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSREJECT</td>
<td>Reject position for inserted bills</td>
</tr>
<tr>
<td>WFS_CDM_POSTOP</td>
<td>Top output position.</td>
</tr>
<tr>
<td>WFS_CDM_POSBOTTOM</td>
<td>Bottom output position.</td>
</tr>
</tbody>
</table>

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_PSUNKNOWN</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 in operative due to a hardware failure or media jam.</td>
</tr>
<tr>
<td>WFS_CDM_TPUNKNOWN</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_TPSTATEMPTY</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>
</tbody>
</table>
Due to a hardware error or other condition it is not known whether there are items on the transport.

The device is not capable of reporting whether items are on the transport.

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

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

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

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 LPWFS_CDMCAPS  lpCaps;

typedef struct _wfs_cdm_caps
{
  WORD     wClass;
  WORD     fwType;
  WORD     wMaxDispenseItems;
  WORD     wCashInMaxBills;
  WORD     wMaxCoins;
  BOOL     bCompound;
  BOOL     bShutter;
  BOOL     bShutterControl;
  BOOL     bRetract;
  WORD     fwRetractAreas;
  WORD     fwRetractTransportActions;
  WORD     fwRetractStackerActions;
  BOOL     bSafeDoor;
  BOOL     bCashInSafeDoor;
  BOOL     bBills;
  BOOL     bCashInBills;
  BOOL     bCoins;
  BOOL     bCylinders;
  BOOL     bCashBox;
  BOOL     bCashIn;
  BOOL     bRefill;
  BOOL     bAutoDeposit;
  BOOL     bVandalCheck;
  BOOL     bIntermediateStacker;
  BOOL     bCashInIntermediateStacker;
  BOOL     bBillsTakenSensor;
  BOOL     bItemsTakenSensor;
  WORD     fwMoveItems;
  WORD     fwExchangeType;
  LPSTR    lpszExtra;
} WFS_CDMCAPS, * LPWFS_CDMCAPS;

Specifies the service class. Value is:
WFS_SERVICE_CLASS_CDM

Supplies the type of CDM as one of the following values:
**Value** | **Meaning**
---|---
WFS_CDM_TELLERBILL | The CDM is a Teller Bill Dispenser.  
WFS_CDM_SELF_SERVICEBILL | The CDM is a Self Service Bill Dispenser.  
WFS_CDM_TELLERCOIN | The CDM is a Teller Coin Dispenser.  
WFS_CDM_SELF_SERVICECOIN | The CDM is a Self Service Coin Dispenser.  

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

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

**bShutter**
Specifies whether or not the commands WFS_CDM_OPEN_SHUTTER and WFS_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_CDM_OPEN_SHUTTER and the WFS_CDM_CLOSE_SHUTTER commands. This field is always set to TRUE if the device has no shutter. This field applies to all shutters and all output positions.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| WFS_CDM_RA_RETRACT | The items may be retracted to the retract cash unit.  
| WFS_CDM_RA_TRANSPORT | The items may be retracted to the transport.  
| WFS_CDM_RA_STACKER | The items may be retracted to the intermediate stacker.  
| WFS_CDM_RA_REJECT | The items may be retracted to the reject cash unit.  
| WFS_CDM_RA_NOTSUPP | The CDM does not have the ability to retract.  

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| WFS_CDM_PRESENT | The items may be presented.  
| WFS_CDM_RETRACT | The items may be retracted to a retract cash unit.  
| WFS_CDM_REJECT | The items may be rejected to a reject bin.  
| WFS_CDM_NOTSUPP | The CDM does not have the ability to retract from the transport.  

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

<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 rejected to a reject bin.</td>
</tr>
<tr>
<td>WFS_CDM_NOTSUPP</td>
<td>The CDM does not have the ability to retract from the stacker.</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 parameter 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_ITEMS_TAKEN event. If set to FALSE this event is not generated. This field applies to all output positions.

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

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

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

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

3.4.3. WFS_INF_CDM_CASH_UNIT_INFO

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 physical cash unit currently present in the device, information about the missing cash unit will still be returned in the lpList field of the output parameter. The status of the cash unit will be reported as WFS_CDM_STATCUMISSING.

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

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

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

lpCashUnitName
usType
cCurrencyID
ulValues

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

Counts
The values of the following fields of the WFSCDMCASHUNIT and WFSCDMPHCU structures:

ulCount
ulRejectCount

are software counts and therefore may not represent the actual number of items in the cash unit.

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

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

The application can check if the device has this capability by querying the bHardwareSensors field of the physical cash unit structure. If any of the physical cash units associated with the logical cash unit have this capability, then threshold events based on hardware sensors may be triggered.
In the situation where the cash unit is associated with multiple physical cash units, if the service provider has the capability, the service event WFS_SRVE_CDM_CASHUNITINFOCHANGED 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_CASHUNITITHTHRESHOLD event will be generated.

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

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

**Input Param**
None.

**Output Param**
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 0. In other commands that use this structure, and that relate to individual tellers (i.e., WFS_CMD_CDM_SET_CASH_UNIT_INFO, WFS_CMD_CDM_START_EXCHANGE, WFS_CMD_CDM_END_EXCHANGE), this field contains the appropriate teller ID value.

- **usCount**
  Specifies the number of cash unit structures returned.

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

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

- **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</td>
</tr>
<tr>
<td></td>
<td>is missing.</td>
</tr>
<tr>
<td>WFS_CDM_TYPEREJECTCASSETTE</td>
<td>Reject cash unit. of the cash dispenser.</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</td>
</tr>
<tr>
<td></td>
<td>material.</td>
</tr>
<tr>
<td>WFS_CDM_TYPEDOCUMENT</td>
<td>Cash unit containing documents.</td>
</tr>
<tr>
<td>WFS_CDM_TYPECOUPON</td>
<td>Replenishment cash unit.</td>
</tr>
<tr>
<td>WFS_CDM_TYPECASHIN</td>
<td>Recycling cash unit. This unit is only</td>
</tr>
<tr>
<td></td>
<td>present when the device is a compound</td>
</tr>
<tr>
<td></td>
<td>device with a CIM.</td>
</tr>
</tbody>
</table>

**cUnitID**
The Cash Unit Identifier.

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

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

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

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

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

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

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

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

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

\textbf{ulMinimum}

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

\textbf{ulMaximum}

This field is only applicable to Retract and Reject Cash Units. When \textit{ulCount} reaches this value the threshold event WFS_USRE_CDM_CASHUNITTHRESHOLD will be generated. If this value is 0 hardware sensors in the device will trigger the threshold event.

\textbf{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.

\textbf{usStatus}

Supplies the status of the 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 (i.e. nearing the threshold defined by \textit{ulMaximum}).</td>
</tr>
<tr>
<td>WFS_CDM_STATCULOW</td>
<td>The cash unit is almost empty (i.e. nearing the threshold defined by \textit{ulMinimum}).</td>
</tr>
<tr>
<td>WFS_CDM_STATCUEMPTY</td>
<td>The cash unit is empty.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUINOP</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. The cash unit has not been calibrated.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUMANIP</td>
<td>The cash unit has been changed when the device was not in the exchange state. This cash unit cannot be dispensed from.</td>
</tr>
</tbody>
</table>

\textbf{usNumPhysicalCUs}

The number of physical cash unit structures returned in the following \textit{lppPhysical} array. This number must be at least 1.

\textbf{lppPhysical}

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

\begin{verbatim}
typedef struct _wfs_cdm_physicalcu
{
    LPSTR lpPhysicalPositionName;
    CHAR cUnitID[5];
    ULONG ulInitialCount;
    ULONG ulCount;
    ULONG ulRejectCount;
    ULONG ulMaximum;
    USHORT usPStatus;
    BOOL bHardwareSensor;
} WFS_CDMPHCU, * LPWFS_CDMPHCU;
\end{verbatim}

\textbf{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.

\textbf{cUnitID}

A 5 character array uniquely identifying the physical cash unit.

\textbf{ulInitialCount}

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

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

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

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

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

<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_STATCUIHIGH</td>
<td>The cash unit is almost full (threshold defined by ( ulMaximum )).</td>
</tr>
<tr>
<td>WFS_CDM_STATCULOW</td>
<td>The cash unit is almost empty (threshold defined by ( ulMinimum )).</td>
</tr>
<tr>
<td>WFS_CDM_STATCUEEMPTY</td>
<td>The cash unit is empty.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUINOP</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. The cash unit has not been calibrated.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUMANIP</td>
<td>The cash unit has been changed when the device was not in the exchange state. This cash unit cannot be dispensed from.</td>
</tr>
</tbody>
</table>

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

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

**Comments**
None.

### 3.4.4. 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

```
LPWFSFSCDMTELLERINFO lpTellerInfo;
typedef struct _wfs_cdm_teller_info
{
    USHORT usTellerID;
    CHAR cCurrencyID[3];
} WFSFSCDMTELLERINFO, *LPWFSFSCDMTELLERINFO;
```

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

**cCurrencyID**
Three character ISO format currency identifier [Ref 2]
This parameter can be an array of three ASCII 0x20h characters. In this case information on all currencies will be returned.

Output Param

```
LPWFSFSCDMTELLERDETAILS * lpTellerDetails;
```

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

```
typedef struct _wfs_cdm_teller_details
{
    USHORT usTellerID;
    ULONG ulInputPosition;
    WORD fwOutputPosition;
    LPWFSFSCDMTELLERTOTALS* lppTellerTotals;
} WFSFSCDMTELLERDETAILS, *LPWFSFSCDMTELLERDETAILS;
```

**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>
<tr>
<td>WFS_CDM_POSREAR</td>
<td>Rear position is assigned to the Teller.</td>
</tr>
</tbody>
</table>

**lppTellerTotals**
Pointer to a null-terminated array of pointers to teller total structures.
typedef struct _wfs_cdm_teller_totals
{
    USHORT           usTellerID
    CHAR   cCurrencyID[3];
    ULONG    ulItemsReceived;
    ULONG    ulItemsDispensed;
    ULONG    ulCoinsReceived;
    ULONG    ulCoinsDispensed;
    ULONG    ulCashBoxReceived;
    ULONG    ulCashBoxDispensed;
} WFSCDMTELLERTOTALS, *LPWFSCDMTELLERTOTALS

\textit{cCurrencyID}

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

\textit{ulItemsReceived}

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

\textit{ulItemsDispensed}

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

\textit{ulCoinsReceived}

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

\textit{ulCoinsDispensed}

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

\textit{ulCashBoxReceived}

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

\textit{ulCashBoxDispensed}

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

\textbf{Error Codes}

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

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

\textbf{Comments}

None.

\textbf{3.4.5. WFS_INF_CDM_CURRENCY_EXP}

\textbf{Description}

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

\textbf{3.4.6. WFS_INF_CDM_MIX_TYPES}

\textbf{Description}

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

\textbf{Input Param}

None.
Output Param  
`LPWFSCDMMIXTYPE * lppMixTypes;`

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

```c
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 or table. Individual vendor-defined mix algorithms are defined above hexadecimal 7FFF. Mix algorithms which are provided by the service provider are in the range hexadecimal 8000 - 8999. Application defined mix algorithms start at hexadecimal 9000. All numbers below 8000 hexadecimal are reserved.

Predefined values are:

<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 value items are dispensed.</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.

### 3.4.7. WFS_INF_CDM_MIX_TABLE

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

**Input Param**
`LPUSHORT lpusMixNumber;`

**lpusMixNumber**
Points to the number of the requested house mix table.
Output Param  

LPWFSFCDMIXTABLE  lpMixTable;

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

usMixNumber  
Number identifying the house mix table.

lpszName  
Points to the name of the table.

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

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

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

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

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

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

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

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

<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>

3.4.8. WFS_INF_CDM_PRESENT_STATUS  

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

This value is persistent and is valid until the next time an attempt is made to present items to the customer.
Input Param  
LPWORD lpfwPosition;

\textit{lpfwPosition}  
Specifies 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;

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

\textit{wPresentState}  
Supplies the \textit{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 did not have 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>

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

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

Comments  
None.

\section*{3.5. \textit{Execute Commands moved to the CIM specification}}

3.5.1. WFS\_CMD\_CDM\_CASH\_IN

3.5.2. WFS\_CMD\_CDM\_CASH\_IN\_START

3.5.3. WFS\_CMD\_CDM\_CASH\_IN\_END

3.5.4. WFS\_CMD\_CDM\_CASH\_IN\_ROLLBACK
3.6. **New Execute Commands**

3.6.1. **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**

```c
LPWFSCDMPHYSICALCU lpPhysicalCU;
```

Pointer to a WFSCDMPHYSICALCU structure:

```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*
A value specifying 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_POSREJECT</td>
<td>Reject bin is used as output location.</td>
</tr>
</tbody>
</table>

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

**Output Param**

```c
LPWFSCDMCOUNT lpCount;
```

Pointer to a WFSCDMCOUNT structure:

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

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

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

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

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

<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_STATCUTFULL</td>
<td>The cash unit is full.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUHIGH</td>
<td>The cash unit is almost full (threshold defined by <em>ulMaximum</em>).</td>
</tr>
<tr>
<td>WFS_CDM_STATCULOW</td>
<td>The cash unit is almost empty (threshold defined by <em>ulMinimum</em>).</td>
</tr>
<tr>
<td>WFS_CDM_STATCUEMPTY</td>
<td>The cash unit is empty.</td>
</tr>
<tr>
<td>WFS_CDM_STATCUINOP</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 changed when the device was not in the exchange state. This cash 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.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM service 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</td>
</tr>
</tbody>
</table>
3.6.2. WFS_CMD_CDM_RESET

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

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

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

**Input Param**

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

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

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

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

<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>

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

**Output Param**
None.

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

<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.</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>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in the exchange state.</td>
</tr>
</tbody>
</table>

```c
output location before the operation can continue.
```
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_MEDIADETECTED</td>
<td>Media has been found in the device.</td>
</tr>
</tbody>
</table>

Comments  None.

3.6.3. WFS_CMD_CDM_TEST_CASH_UNITS

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

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

Input Param  LPWFS_CDMITEMPOSITION lpPosition

Specifies where items dispensed as a result of this command should be moved to. For a description of the WFS_CDMITEMPOSITION structure see section WFS_CMD_CDM_RESET.

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

Output Param  LPWFS_CDMCUINFO lpCUInfo;

The WFS_CDMCUINFO 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 or the cash unit could not be tested. 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_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 service is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_PRERRORNOREITEMS</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>
<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 has failed the test or a cash unit could not be tested because it is inoperative, empty or locked.</td>
</tr>
<tr>
<td>WFS_SRVE_CDM_ITEMSTAKEN</td>
<td>The items presented have been removed by the user.</td>
</tr>
</tbody>
</table>

3.7. **Execute Command removed**

3.7.1. **WFS_CMD_CDM_CHECK_VANDALISM**

3.7.2. **WFS_CMD_CDM_SET_TELLER_POSITIONS**

All teller information can now be set with the WFS_CMD_CDM_SET_TELLER_INFO command.

3.8. **Changes to existing Execute Commands**

3.8.1. **WFS_CMD_CDM_DENOMINATE**

**Description**

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

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

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

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

This command can be used in four different ways:

1. In order to check that it is possible to dispense a given denomination. The input parameters to the command are currency and denomination, with a mix number of WFS_CDM_INDIVIDUAL and an amount of 0. If items of differing currencies are to be dispensed then the currency field should be an array of three ASCII 0x20h characters.

2. In order to validate that a given amount matches a given denomination and that it is possible to dispense the denomination. The input parameters to the command should be amount and denomination, with a mix number of WFS_CDM_INDIVIDUAL.

3. In order to obtain a denomination of a given amount. The input parameters supplied should be amount, currency and mix number.

4. In order to complete a partial denomination of a given amount. In this case the input parameters to the command should be currency, amount, mix number and either a partially specified denomination or a minimum amount from the cash box. A completed denomination...
is returned. \texttt{ulCashBox} of the denomination structure may be updated as a result of this command.

\textbf{Input Param} \quad \texttt{LPWFS\_CDM\_DENOMINATE lpDenominate;}

\begin{verbatim}
typedef struct _wfs_cdm_denominate
{
    USHORT usTellerID;
    USHORT usMixNumber;
    LPWFS\_CDM\_DENOMINATE lpDenomination;
} WFS\_CDM\_DENOMINATE, *LPWFS\_CDM\_DENOMINATE;
\end{verbatim}

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

\textit{usMixNumber}  
Mix algorithm or house mix table to be used.

\textit{lpDenomination}  
Pointer to a WFS\_CDM\_DENOMINATION structure, describing the contents of the denomination operation.

\begin{verbatim}
typedef struct _wfs_cdm_denomination
{
    CHAR cCurrencyID[3];
    ULONG ulAmount;
    USHORT usCount;
    LPULONG lpulValues;
    ULONG ulCashBox;
} WFS\_CDM\_DENOMINATION, *LPWFS\_CDM\_DENOMINATION;
\end{verbatim}

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

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

\textit{usCount}  
The size of the \textit{lpulValues} list. \textit{This \textit{usCount} is the same as the \textit{usCount} returned from the last WFS\_INF\_CDM\_CASH\_UNIT\_INFO command or set by the last WFS\_CMD\_CDM\_SET\_CASH\_UNIT\_INFO or WFS\_CMD\_CDM\_END\_EXCHANGE commands. If this value is not required because a mix algorithm is used then the \textit{usCount} can be set to 0.}

If the application passes in an invalid \textit{usCount} the service provider should return a WFS\_ERR\_INVALID\_DATA return code.

\textit{lpulValues}  
Pointer to an array of ULONGs. \textit{This list specifies the number of items to take from each of the cash units. This list corresponds to the array of cash unit structures returned to the last WFS\_INF\_CDM\_CASH\_UNIT\_INFO command or set by the last WFS\_CMD\_CDM\_SET\_CASH\_UNIT\_INFO or WFS\_CMD\_CDM\_END\_EXCHANGE commands. The first value in the array is related to the cash structure with the index number 1.}

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

If the Application does not wish to specify a denomination, it should set the \textit{lpulValues} pointer to NULL.

\textit{ulCashBox}  
Only applies to Teller CDM devices. \textit{Amount to be paid from the teller's cash box.}
Output Param  LPWFS_CDM_DENOMINATION  lpDenomination;
For a description see the input structure.

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

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

<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 cCurrency field of the input parameter.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDTELLERID</td>
<td>Invalid Teller ID.</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 cashbox and denomination was greater than the amount specified.</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_TOOMANYBILLS</td>
<td>The request would require too many bills to be dispensed.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_TOOMANYCOINS</td>
<td>The request would require too many coins to be dispensed.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_TOOMANYITEMS</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 WFS_CMD_CDM_START_EXCHANGE)</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOCASHBOXPRESENT</td>
<td>Cash box amount needed, however teller is not assigned a Cash Box.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_AMOUNTNOTINMIXTABLE</td>
<td>A mix table is being used to determine the denomination but the amount specified for the denomination is not in the mix table.</td>
</tr>
</tbody>
</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.

3.8.2. WFS_CMD_CDM_DISPENSE

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

1. The input parameters to the command are amount, currency and denomination. The mix number is WFS_CDM_INDIVIDUAL. In this case, the denomination is checked for validity and, if valid, is dispensed.

2. The input parameters are amount, currency and mix number. In this case the amount is denominated and, if this succeeds, the items are dispensed.

3. If the amount is 0, but the currency and the denomination are supplied with a mix number of WFS_CDM_INDIVIDUAL, the denomination is checked for validity and, if valid, is dispensed.

4. The command will calculate a partial denomination of a given amount and dispense the complete denomination. In this case the input parameters to the command should be currency, amount, mix number and either a partially specified denomination or a minimum amount from the cash box. The cashbox amount may be updated as a result of this command.

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

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

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

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

**Input Param**

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

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

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

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

<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>
</tbody>
</table>
**WFS_CDM_POSTOP** Present items to the top output position.

**WFS_CDM_POSTBOTTOM** Present items to the bottom output position.

**WFS_CDM_POSFRONT** Present items to the front output position.

**WFS_CDM_POSREAR** Present items to the rear output position.

**bPresent**
If this field is set to TRUE then the items will be moved to the exit slot, if it is FALSE the items will be moved to an intermediate stacker.

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

**Output Param**

```c
LPWFSCDMDENOMINATION lpDenomination;
```
For the WFSCDMDENOMINATION structure specification see the definition of the command WFS_CMD_CDM_DENOMINATE.

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

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

**Error Codes**

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

<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 cCurrency field of the input parameter.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDTELLERID</td>
<td>Invalid Teller ID.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>There is a problem with a cash unit. The 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_UNSUPOPOSITION</td>
<td>The specified output position is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_POSITIONLOCKED</td>
<td>The output position is locked.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_SAFEDOOROPEN</td>
<td>The safe door is open.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_TOOMANYBILLS</td>
<td>The request would require too many bills to be dispensed.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_TOOMANYCOINS</td>
<td>The request would require too many coins to be dispensed.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOCASHBOXPRESENT</td>
<td>Cash box amount needed, however teller is not assigned a Cash Box.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_AMOUNTNOTINMIXTABLE</td>
<td>A mix table is being used to determine the denomination but the amount specified for the denomination is not in the mix table.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_ITEMSNOTTAKEN</td>
<td>Items have not been taken during a sub-dispense operation. This error occurs if a hardware timeout expires.</td>
</tr>
</tbody>
</table>
CWA 14050-19:2000

WFS_ERR_CDM_ITEMSLEFT: Items have been left in the transport or exit slot as a result of a prior Dispense, Present or Recycler Cash-In operation.

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

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

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

<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_INCOMPLETEDISPENSE</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>A notes detection error has occurred.</td>
</tr>
</tbody>
</table>

3.8.3. WFS_CMD_CDM_PRESENT

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

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

When this command successfully completes the items are in customer access.
**Input Param**  LPWORD lpfwPosition

*fwPosition*

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

<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 service 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 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>
</tbody>
</table>

**Comments**  None.

### 3.8.4. WFS_CMD_CDM_REJECT

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

**Input Param**  None.

**Output Param**  None.
Error Codes

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>The 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 service 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>
</tbody>
</table>

Comments

None.

3.8.5. WFS_CMD_CDM_RETRACT

Description

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

Input Param

LPWFS_CMD_RETRACT lpRetract;

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

fwOutputPosition

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

<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>
</tbody>
</table>

usIndex

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

**Output Param** None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>The retract 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 to retract.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_SHUTTERNOTCLOSED</td>
<td>The shutter failed to close.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_ITEMSTAKEN</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDRETRACT</td>
<td>Retract function is invalid for this system.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDRETRACTPOSITION</td>
<td>The usIndex is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOTRETRACTAREA</td>
<td>The retract area specified in usRetractArea is not supported.</td>
</tr>
</tbody>
</table>

**Events** In addition to the generic events defined in [Ref. 1], the following additional 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 the retract or reject cash unit.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_CASHUNITERROR</td>
<td>An error occurred while attempting to retract to the retract or reject cash unit.</td>
</tr>
</tbody>
</table>

**Comments** None.

### 3.8.6. WFS_CMD_CDM_OPEN_SHUTTER

**Description** This command is used only for ATMs. This command opens the shutter.

**Input Param**

| LPWORD lpfwPosition;                        |

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

<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.

3.8.7. WFS_CMD_CDM_CLOSE_SHUTTER

Description
This command is used only for ATMs. This command closes the shutter.

Input Param
LPWORD lpfwPosition;

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

<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.

3.8.8. WFS_CMD_CDM_SET_TELLER_INFO

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

Input Param
LPWFSxCDMTELLERUPDATE lpTellerUpdate
typedef struct _wfs_cdm_teller_update {
    USHORT usAction;
    LPWFSxCDMTELLERDETAILS lpTellerDetails;
} WFSxCDMTELLERUPDATE *LPWFSxCDMTELLERUPDATE;
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>

For a specification of the struct WFSCDMTELLERDETAILS please refer to the WFS_INF_CDM_TELLER_INFO command.

Output Param
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_INVALIDCURRENCY</td>
<td>The specified currency not currently available.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDTELLERID</td>
<td>The Teller ID is invalid.</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 current 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.

3.8.9. WFS_CMD_CDM_SET_CASH_UNIT_INFO

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

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

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

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

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

Any other changes must be performed via an exchange operation.

If the fields ulCount and ulRejectCount of lppPhysical are set to 0 by this command, the application is indicating that it does not wish counts to be maintained for the physical cash units. Counts on the logical cash units will still be maintained and can be used by the application. If the physical counts are set by this command then the logical count will be the sum of the physical counts and any value sent as a logical count will be ignored.
Input Param
LPWFS_CDMCUINFO 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_CASHUNITERROR</td>
<td>A cash unit specified caused a problem. A WFS_EXEE_CDM_CASHUNITERROR execute event is posted with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDTELLERID</td>
<td>Invalid Teller ID.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDCASHUNIT</td>
<td>Invalid cash unit ID.</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 threshold condition has been reached in one of the cash units.</td>
</tr>
<tr>
<td>WFS_SRVE_CDM_CASHUNITINFOCHANGED</td>
<td>A cash unit was updated as a result of this command.</td>
</tr>
</tbody>
</table>

Comments
None.

3.8.10. WFS_CMD_CDM_START_EXCHANGE

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

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

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

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

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

- WFS_CMD_CDM_END_EXCHANGE
- Any WFS[Async]Get Info commands
- WFS_CLOSE – this will end the exchange state
- WFS_CMD_CDM_SET_MIX_TABLE

Any other commands will result in the error WFS_ERR_CDM_EXCHANGE_ACTIVE being generated

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

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

CDM

(Lock)

WFS_CMD_CDM_START_EXCHANGE

...exchange action...

WFS_CMD_CDM_END_EXCHANGE

(Unlock)

CIM

(Lock)

WFS_CMD_CIM_START_EXCHANGE

...exchange action...

WFS_CMD_CIM_END_EXCHANGE

(Unlock)

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

Input Param

LPWFS_CDMSTARTEX 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 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 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

LPWFS_CDMCUINFO 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 CDM.</td>
</tr>
</tbody>
</table>
WFS_ERR_CDM_CASHUNITERROR  An error occurred with a cash unit while performing the exchange operation. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.

WFS_ERR_CDM_EXCHANGEACTIVE  The CDM is already in an exchange state.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_EXEE_CDM_CASHUNITERROR</td>
<td>An error occurred while performing the exchange.</td>
</tr>
<tr>
<td>WFS_EXEE_CDM_NOTEERROR</td>
<td>A note detection error has occurred.</td>
</tr>
</tbody>
</table>

Comments  None.

3.8.11. WFS_CMD_CDM_END_EXCHANGE

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

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

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

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

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

The values set by this command are persistent.

Input Param  
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. Otherwise the parameter must contain the complete list of cash unit structures, not just the ones that have changed.

Output Param  None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_INVALIDTELLERID</td>
<td>Invalid Teller ID.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>This error is returned if there is a problem with the values set for a cash unit. A WFS_EXEE_CDM_CASHUNITERROR event will be posted with the details.</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>The values of the cash unit structures are incorrect. The cash unit structure that is incorrect is returned as a parameter on this event.</td>
</tr>
</tbody>
</table>
3.8.12. 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.

3.8.13. WFS_CMD_CDM_CALIBRATE_CASH_UNIT

**Description**

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

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

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

**Input Param**

```c
typedef struct _wfs_cdm_calibrate
{
    USHORT usNumber;
    USHORT usNumOfBills;
    LPWFSITEMPOSITION *lpPosition;
} WFSITEMPOSITION, *LPWFSITEMPOSITION;
```

- **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 WFSITEMPOSITION structure see Section WFS_CMD_CDM_RESET.

**Output Param**

```c
LPWFSITEMPOSITION lpCalibrateOut;
```

The WFSITEMPOSITION structure is defined in the Input Param section.

- **usNumber**
  The logical number of cash unit which has been calibrated.

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

- **lpPosition**
  Specifies where the items were moved to during the calibration process.
### Error Codes
In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

<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.</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>
</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_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>
</tbody>
</table>

### Comments
None.

### 3.8.14. 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 by the service provider will attempt to denominate or dispense the first amount in the table. If this does not succeed (e.g. because of a cash unit failure) the service provider will attempt to denominate or dispense the next amount in the table. The service provider can only dispense amounts which are explicitly mentioned in the mix table.

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

The values set by this command are persistent.

**Input Param**
LPWFS_CDM_MIXTABLE lpMixTable;

The structure WFS_CDM_MIXTABLE 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_EXCHANGEACTIVE</td>
<td>The CDM is in an exchange state.</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.

### 3.9. Events moved to the CIM specification

#### 3.9.1. WFS_EXEE_CDM_INPUTREFUSE
3.10. **New Events**

### 3.10.1. WFS_SRVE_CDM_COUNTS_CHANGED

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

**Event Param**
```c
typedef struct _wfs_cdm_counts_changed
{
    USHORT usCount;
    USHORT * lpusCUNumList;
} WFS_CDM_COUNTS_CHANGED, *LPWFS_CDM_COUNTS_CHANGED;
```

- **usCount**
The size of lpusCUNumList.

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

**Comments**
None.

### 3.10.2. WFS_EXEE_CDM_INCOMPLETEDISPENSE

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

**Event Param**
```c
LPWFS_CDM_DENOMINATION lpDenomination;
```

- **lpDenomination**
The `WFS_CDM_DENOMINATION` 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.

### 3.10.3. WFS_EXEE_CDM_NOTEERROR

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

**Event Param**
```c
LPUSHORT lpusReason;
```

- **lpusReason**
Specifies 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>A bill counting error has occurred.</td>
</tr>
<tr>
<td>WFS_CDM_NOTESTOOCLOSE</td>
<td>Notes have been detected as being too close.</td>
</tr>
</tbody>
</table>

**Comments**
None.
3.10.4. WFS_SRVE_CDM_ITEMSPRESENTED

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

Event Param: None.

Comments: None.

3.10.5. 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: LPWFS_CMDITEMPOSITION * lpItemPosition;
For a description of this parameter see WFS_CMD_CDM_RESET (section 3.6.2)

Comments: None.

3.11. Events removed

3.11.1. WFS_SRVE_CDM_SAFEDOORLOCKED

3.12. Changes to existing Events

3.12.1. WFS_SRVE_CDM_ITEMSTAKEN (former WFS_SRVE_CDM_BILLSTAKEN)

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

Event Param: LPWORD lpfwPosition;

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

<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>

Comments: None.

3.12.2. WFS_SRVE_CDM_SAFEDOOROPEN

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

Event Param: None.

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

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

- WFS_CMD_CDM_SET_CASH_UNIT_INFO
- WFS_CMD_CDM_END_EXCHANGE
- WFS_CMD_CDM_CALIBRATE_CASH_UNIT

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

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

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

Event Param LPWFS_CDM_CASHUNIT 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.

3.12.5. 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.

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

Event Param LPULONG lpulDelay;

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

Comments None.
3.12.7. **WFS_EXEE_CDM_STARTDISPENSE**

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

**Event Param**
LPREQUESTID lpReqID;

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

**Comments**
None.

3.12.8. **WFS_EXEE_CDM_CASHUNITERROR**

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

**Event Param**
LPWFS_CDM_CU_ERROR lpCashUnitError;

typedef struct _wfs_cdm_cu_error
{
    WORD wFailure;
    LPWFS_CDM_CASHUNIT lpCashUnit;
} WFS_CDM_CU_ERROR, *LPWFS_CDM_CU_ERROR;

`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 ID is invalid.</td>
</tr>
<tr>
<td>WFS_CDM_CASHUNITCONFIG</td>
<td>An attempt has been made to change the settings of a self-configuring cash unit.</td>
</tr>
</tbody>
</table>

`lpCashUnit`
Pointer to the cash unit structure that caused the problem. The WFS_CDM_CASHUNIT 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.

3.12.9. **WFS_EXEE_CDM_PARTIALDISPENSE**

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

**Event Param**
LPUSHORT lpusDispNum;

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

**Comments**
None.
3.12.10. 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
LPWFSCTMDENOMINATION lpDenomination;

lpDenomination
The WFSCDM DENOMINATION 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.

3.13. New Sections

3.13.1. Sub-Dispensing Command Flow

3.13.2. Rules for Cash Unit Exchange
4. **CIM**

4.1. **New Info Commands**

4.1.1. **WFS_INF_CIM_BANKNOTE_TYPES**

**Description**
This command is used to obtain information about the banknote types that can be detected by the banknote reader.

**Input Param**
None.

**Output Param**

```
LPWFSIMNOTETYPELIST lpNoteTypeList;
```

```c
typedef struct _wfs_cim_note_type_list
{
    USHORT usNumOfNoteTypes;
    LPWFSIMNOTETYPE *lppNoteTypes;
} WFSIMNOTETYPELIST, *LPWFSIMNOTETYPELIST;
```

- **usNumOfNoteTypes**
  Number of banknote types the banknote reader supports, i.e. the size of the `lppNoteTypes` list.

- **lppNoteTypes**
  List of banknote types the banknote reader supports. A pointer to an array of pointers to WFSIMNOTETYPE structures:

```c
typedef struct _wfs_cim_note_type
{
    USHORT usNoteID;
    CHAR cCurrencyID[3];
    ULONG ulValues;
    USHORT usRelease;
    BOOL bConfigured;
} WFSIMNOTETYPE, *LPWFSIMNOTETYPE;
```

- **usNoteID**
  Identification of note type.

- **cCurrencyID**
  Currency ID in ISO 4217 format [see Ref. 2].

- **ulValues**
  The value of a single item expressed in minimum dispense units.

- **usRelease**
  The release of the banknote type. The higher this number, the newer the release. Zero means that there is only one release of that banknote type. This value has not been standardised and therefore a release number of the same banknote will not necessarily have the same value in different systems.

- **bConfigured**
  Specifies whether or not the banknote reader recognizes this note type. If TRUE the banknote reader will accept this note type during a Cash-In operation, if FALSE the banknote reader will refuse this note type.

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

**Comments**
None.

4.1.2. **WFS_INF_CIM_CASH_IN_STATUS**

**Description**
This command is used to get information about the status of the last cash in transaction. This value is persistent and is valid until the next WFS_CMD_CIM_CASH_IN_START.

**Input Param**
None.
Output Param  LPWFSICMCASHINSTATUS lpStatus;

typedef struct _wfs_cim_cash_in_status
{
  WORD wStatus;
  USHORT usNumOfRefused;
  LPWFSICMNOTENUMBERLIST lpNoteNumberList;
  LPSTR lpszExtra;
} WFSCIMCASHINSTATUS, *LPWFSCIMCASHINSTATUS;

wStatus
Status of the Cash-In transaction. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_CIOK</td>
<td>The cash in transaction is complete.</td>
</tr>
<tr>
<td>WFS_CIM_CIROLLBACK</td>
<td>The cash in transaction was rolled back.</td>
</tr>
<tr>
<td>WFS_CIM_CIACTIVE</td>
<td>There is a cash in transaction active.</td>
</tr>
<tr>
<td>WFS_CIM_CIRETRACT</td>
<td>The cash-in transaction ended with the items being retracted.</td>
</tr>
<tr>
<td>WFS_CIM_CIUNKNOWN</td>
<td>The state of the cash in transaction is unknown.</td>
</tr>
</tbody>
</table>

usNumOfRefused
Specifies the number of items refused during the Cash-In transaction period.

lpNoteNumberList
List of banknote types that were inserted, identified and accepted during the Cash-In transaction period. If notes have been rolled back they will be included in this list. For a description of the WFSICMNOTENUMBERLIST structure see the definition of the command WFS_INF_CIM_CASH_UNIT_INFO.

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

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

Comments
None.

4.2. Changes to Info Commands which previously existed in the CDM

4.2.1. WFS_INF_CIM_STATUS

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

Input Param
None.

Output Param  LPWFSICMSTATUS lpStatus;

typedef struct _wfs_cim_status
{
  WORD fwDevice;
  WORD fwSafeDoor;
  WORD fwCashInSafeDoor;
  WORD fwDispenser;
  WORD fwAcceptory;
  WORD fwIntermediateStacker;
  WORD fwStackerItems;
  WORD fwBanknoteReader;
  BOOL bDropBox;
  LPWFSICMINPOS * lppPositions;
  LPSTR lpszExtra;
} WFSCIMSTATUS, *LPWFSCIMSTATUS;
fwDevice
Supplies the state of the CIM. However, a fwDevice status of WFS_CIM_DEVONLINE does not necessarily imply that accepting can take place: the value of the fwAcceptor 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 CIM will have one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_DEVONLINE</td>
<td>The device is online. This is returned when the acceptor is present and operational.</td>
</tr>
<tr>
<td>WFS_CIM_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_CIM_DEVPOWEROFF</td>
<td>The device is powered off or physically not connected.</td>
</tr>
<tr>
<td>WFS_CIM_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_CIM_DEVHWERROR</td>
<td>The device is inoperable due to a hardware error.</td>
</tr>
<tr>
<td>WFS_CIM_DEVUSERERROR</td>
<td>The device is present but a person is preventing proper device operation.</td>
</tr>
<tr>
<td>WFS_CIM_DEVBUSY</td>
<td>The device is busy and unable to process an execute command at this time.</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_CIM_DOORNOTSUPPORTED</td>
<td>Physical device has no safe door or door state reporting is not supported.</td>
</tr>
<tr>
<td>WFS_CIM_DOOROPEN</td>
<td>Safe door is open.</td>
</tr>
<tr>
<td>WFS_CIM_DOORCLOSED</td>
<td>Safe door is closed.</td>
</tr>
<tr>
<td>WFS_CIM_DOORLOCKED</td>
<td>Safe door is closed and locked.</td>
</tr>
<tr>
<td>WFS_CIM_DOORUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the door cannot be determined.</td>
</tr>
</tbody>
</table>

fwAcceptor
Supplies the state of the acceptor cash units as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_ACCOK</td>
<td>All cash units present are in a good state.</td>
</tr>
<tr>
<td>WFS_CIM_ACCCUSTATE</td>
<td>One of the cash units present is in an abnormal state.</td>
</tr>
<tr>
<td></td>
<td>The acceptor is operational, but one or more of the cash units is in a high, full or inoperative condition. Items can still be accepted into at least one of the cash units.</td>
</tr>
<tr>
<td>WFS_CIM_ACCCUSTOP</td>
<td>Due to a cash unit failure accepting is impossible. The acceptor is operational, but no items can be accepted because all of the cash units are in a full or inoperative condition. This state also occurs when a retract cash unit is full or no retract cash unit is present, or an application lock is set on every cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_ACCCUUNKNOWN</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 as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_ISEMPTY</td>
<td>The intermediate stacker is empty.</td>
</tr>
<tr>
<td>WFS_CIM_ISNOTEMPTY</td>
<td>The intermediate stacker is not empty.</td>
</tr>
<tr>
<td>WFS_CIM_ISFULL</td>
<td>The intermediate stacker is full.</td>
</tr>
<tr>
<td>WFS_CIM_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_CIM_ISNOTSUPPORTED</td>
<td>The physical device has no intermediate stacker.</td>
</tr>
</tbody>
</table>
**fwStackerItems**
This field informs the application whether items on the intermediate stacker have been in customer access. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_CUSTOMERACCESS</td>
<td>Items on the intermediate stacker 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 out operation.</td>
</tr>
<tr>
<td>WFS_CIM_NOCUSTOMERACCESS</td>
<td>Items on the intermediate stacker have not been in customer access.</td>
</tr>
<tr>
<td>WFS_CIM_ACCESSUNKNOWN</td>
<td>It is not known if the items on the intermediate stacker have been in customer access.</td>
</tr>
<tr>
<td>WFS_CIM_NOITEMS</td>
<td>There are no items on the intermediate stacker or the physical device has no intermediate stacker.</td>
</tr>
</tbody>
</table>

**fwBanknoteReader**
Supplies the state of the banknote reader as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_BNROK</td>
<td>The banknote reader is in a good state.</td>
</tr>
<tr>
<td>WFS_CIM_BNRINOP</td>
<td>The banknote reader is inoperable.</td>
</tr>
<tr>
<td>WFS_CIM_BNRUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the banknote reader cannot be determined.</td>
</tr>
<tr>
<td>WFS_CIM_BNRNOTSUPPORTED</td>
<td>The physical device has no banknote reader.</td>
</tr>
</tbody>
</table>

**bDropBox**
The drop box is an area with in the CIM where items which have caused a problem during an operation are stored. This field specifies the status of the drop box. TRUE means that some items are stored in the drop box due to a Cash-In transaction which caused a problem. FALSE indicates that the drop box is empty.

**lppPositions**
Pointer to a NULL terminated array of pointers to WFSCIMINPOS structures (one for each supported input or output position):

```c
typedef struct _wfs_cim_inpos {
    WORD fwPosition;
    WORD fwShutter;
    WORD fwPositionStatus;
    WORD fwTransport;
    WORD fwTransportStatus;
} WFSCIMINPOS, * LPWFSCIMINPOS;
```

**fwPosition**
Specifies the input or output position as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>Left input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>Right input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>Center input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>Top input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>Bottom input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRONT</td>
<td>Front input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>Rear input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Rear output position.</td>
</tr>
</tbody>
</table>
### fwShutter
Specifies 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_CIM_SHTCLOSED</td>
<td>The shutter is closed.</td>
</tr>
<tr>
<td>WFS_CIM_SHTOPEN</td>
<td>The shutter is opened.</td>
</tr>
<tr>
<td>WFS_CIM_SHTJAMMED</td>
<td>The shutter is jammed.</td>
</tr>
<tr>
<td>WFS_CIM_SHTUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the shutter cannot be determined.</td>
</tr>
<tr>
<td>WFS_CIM_SHTNOTSUPPORTED</td>
<td>The physical device has no shutter or shutter state reporting is not supported.</td>
</tr>
</tbody>
</table>

### fwPositionStatus
The status of the input or output Position. This field specifies the state of the position as one of the following values:

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

### fwTransport
Specifies 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_CIM_TPOK</td>
<td>The transport is in a good state.</td>
</tr>
<tr>
<td>WFS_CIM_TPINOP</td>
<td>The transport is inoperative due to a hardware failure or media jam.</td>
</tr>
<tr>
<td>WFS_CIM_TPUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the transport cannot be determined.</td>
</tr>
<tr>
<td>WFS_CIM_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 on the transport. If the device is a Cash Recycler it is possible that items will be on the transport due to a previous dispense operation, in which case the status will be WFS_CIM_TPSTATNOTEMPTY. The possible values of this field are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_TPSTATEMPTY</td>
<td>The transport is empty.</td>
</tr>
<tr>
<td>WFS_CIM_TPSTATNOTEMPTY</td>
<td>The transport is not empty, the items have not been in customer access.</td>
</tr>
<tr>
<td>WFS_CIM_TPSTATNOTEMPTYCUST</td>
<td>Items which a customer has had access to are on the transport.</td>
</tr>
<tr>
<td>WFS_CIM_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_CIM_TPSTATNOTSUPPORTED</td>
<td>The device is not capable of reporting whether or not items are on the transport.</td>
</tr>
</tbody>
</table>

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

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

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

Description
This command is used to retrieve the capabilities of the cash acceptor.

Input Param
None.

Output Param

```csharp
typedef struct _wfs_cim_caps {
    WORD wClass;
    WORD fwType;
    WORD wMaxBills;
    WORD wMaxCoins;
    WORD wMaxCashInItems;
    BOOL bCompound;
    BOOL bShutter;
    BOOL bShutterControl;
    BOOL bRetract;
    BOOL bSafeDoor;
    BOOL bCoins;
    BOOL bCylinders;
    BOOL bCashBox;
    BOOL bCashIn;    
    BOOL bRefill;
    BOOL bAutoDeposit;
    BOOL bVandalCheck;
    BOOL bIntermediateStacker;
    WORD fwIntermediateStacker;
    BOOL bItemsTakenSensor;
    BOOL bItemsInsertedSensor;
    WORD fwOutputPositions;
    WORD fwPositions;
    WORD fwExchangeType;
    WORD fwRetractAreas;
    WORD fwRetractTransportActions;
    WORD fwRetractStackerActions;
    LPSTR lpszExtra;
} WFSCIMCAPS, *LPWFSCIMCAPS;
```

**wClass**
Supplies the logical service class. Value is:
WFS_SERVICE_CLASS_CIM

**fwType**
Supplies the type of CIM as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_TELLERBILL</td>
<td>The CIM is a Teller Bill Acceptor.</td>
</tr>
<tr>
<td>WFS_CIM_SELSERVICEBILL</td>
<td>The CIM is a Self Service Bill Acceptor.</td>
</tr>
<tr>
<td>WFS_CIM_TELLERCoin</td>
<td>The CIM is a Teller Coin Acceptor.</td>
</tr>
<tr>
<td>WFS_CIM_SELSERVICECOIN</td>
<td>The CIM is a Self Service Coin Acceptor.</td>
</tr>
</tbody>
</table>

**wMaxCashInItems**
Supplies the maximum number of items that can be accepted in a single cash in operation. Normally reflects hardware limitations of the device.

**bCompound**
Specifies whether or not the logical device is part of a compound physical device and is either TRUE or FALSE.

**bShutter**
If this flag is true explicit shutter control through the commands WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER is 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_CIM_OPEN_SHUTTER and the WFS_CMD_CIM_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.

**bSafedoor**
Specifies whether the WFS_CMD_CIM_OPEN_SAFE_DOOR command is supported.

**bCashBox**
This field is only applicable to CIM types WFS_CIM_TELLERBILL and WFS_CIM_TELLERCOIN. It specifies whether or not the Tellers have been assigned a Cash Box.

**fwIntermediateStacker**
Specifies the number of items the intermediate stacker for Cash-In can hold. Zero means that there is no intermediate stacker for Cash-In available.

**bItemsTakenSensor**
Specifies whether or not the CIM 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_CIM_ITEMS_TAKEN event. If set to FALSE this event is not generated. This field relates to all output positions.

**bItemsInsertedSensor**
Specifies whether the CIM has the ability to detect when items have been inserted by the user. If set to TRUE the service provider generates an accompanying WFS_SRVE_CIM_ITEMSINSERTED event. If set to FALSE this event is not generated. This field relates to all input positions.

**fwPositions**
Specifies the CIM input and 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_CIM_POSINLEFT</td>
<td>Left input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>Right input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>Center input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>Top input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>Bottom input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFROnt</td>
<td>Front input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>Rear input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Rear output position.</td>
</tr>
</tbody>
</table>

**fwExchangeType**
Specifies the type of cash unit exchange operations supported by the CIM. Values are a combination of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_EXBYHAND</td>
<td>The CIM supports manual replenishment either by emptying the cash unit by hand or by replacing the cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_EXTOCASSETTES</td>
<td>The CIM supports moving items from the replenishment cash unit to the bill cash units.</td>
</tr>
<tr>
<td>WFS_CIM_CLEARRECYCLER</td>
<td>The CIM supports the emptying of recycle cash units.</td>
</tr>
<tr>
<td>WFS_CIM_DEPOSITINTO</td>
<td>The CIM supports moving items from the deposit entrance to the bill cash units.</td>
</tr>
</tbody>
</table>
fwRetractAreas
Specifies the areas to which items may be retracted. 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_CIM_RA_RETRACT</td>
<td>Items may be retracted to the retract cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_RA_TRANSPORT</td>
<td>Items may be retracted to the transport.</td>
</tr>
<tr>
<td>WFS_CIM_RA_STACKER</td>
<td>Items may be retracted to the intermediate stacker.</td>
</tr>
<tr>
<td>WFS_CIM_RA_BILLCASSETTES</td>
<td>Items may be retracted to recycle cassettes.</td>
</tr>
<tr>
<td>WFS_CIM_RA_NOTSUPP</td>
<td>The CIM does not have the ability to retract.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_RETRACT</td>
<td>The items may be retracted to a retract cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_NOTSUPP</td>
<td>The CIM does not have the ability to retract from the transport.</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 a retract capability this field will be WFS_CIM_NOTSUPP. Otherwise is will be set to one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_PRESENT</td>
<td>The items may be moved to the exit position.</td>
</tr>
<tr>
<td>WFS_CIM_RETRACT</td>
<td>The items may be retracted to a retract cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_NOTSUPP</td>
<td>The CIM does not have the ability to retract from the stacker.</td>
</tr>
</tbody>
</table>

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

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

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

4.2.3. WFS_INF_CIM_CASH_UNIT_INFO
Description
This command is used to obtain information about the status and contents of the cash in units and recycle units in the CIM.

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

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

It is also possible that multiple logical cash in units or recycle units may be associated with one physical cash unit. This should only occur if the physical cash unit is capable of handling this situation, i.e. if it can store multiple denominations and report meaningful count and replenishment information for each denomination. In this case the information returned in lpCashInfo will again reflect the number of logical cash in units or recycle units in the CIM.
Counts
The value of the ulCount field of the WFSCIMNOTENUMBER structure is a software count and therefore may not represent the actual number of items in the cash unit.

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

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

In the situation where the cash unit is associated with multiple physical cash units, WFS_SRVE_CIM_CASHUNITINFOCHANGED can be generated when each of the physical cash units reaches the threshold. When the final physical cash unit reaches the threshold, the WFS_USRE_CIM_CASHUNITTHRESHOLD event will be generated.

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

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

Input Param None.

Output Param

typedef struct _wfs_cim_cash_info
{
    USHORT usTellerID;
    USHORT usCount;
    LPWFSCDMCASHUNIT * lppList;
    LPWFSCIMCASHIN* lppCashIn;
} WFSCIMCASHINFO, *LPWFSCIMCASHINFO;

usCount
Number of WFSCIMCASHIN structures returned in lppCashIn.

lppCashIn
Pointer to an array of pointers to WFSCIMCASHIN structures:

typedef struct _wfs_cim_cash_in
{
    USHORT usNumber;
    USHORT usType;
    DWORD fwType;
    DWORD fwItemType;
    CHAR cUnitID[5];
    CHAR cCurrencyID[3];
    ULONG ulValues;
    ULONG ulInitialCount;
    ULONG ulCashInCount;
    ULONG ulCount;
    ULONG ulMinimum;
    USHORT usStatus;
    BOOL bDevLock;
    BOOL bAppLock;
    LPSTR lpPhysicalPositionName;
    LPWFSCIMNOTENUMBERLIST lpNoteNumberList;
}
USHORT usNumPhysicalCUs;
LPWFSCIMPHCU * lppPhysical;
LPSTR lpszExtra;

) WFSCIMCASHIN, *LPWFSCIMCASHIN;

**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.

**fwType**
Specifies the type of cash unit takes one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_TYPERECYCLING</td>
<td>Recycle cash unit. This type of cash unit is present only when the device is a Cash Recycler. It can be used for cash dispensing.</td>
</tr>
<tr>
<td>WFS_CIM_TYPECASHIN</td>
<td>Cash-In cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_TYPEREPCONTAINER</td>
<td>Replenishment container. A cash unit can be refilled from a replenishment container.</td>
</tr>
<tr>
<td>WFS_CIM_TYPERETRACTCASSETTE</td>
<td>Retract cash unit.</td>
</tr>
</tbody>
</table>

**fwItemType**
Specifies the type of items the Cash Unit takes as a combination of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_CITYPALL</td>
<td>The cash in unit takes all banknote types.</td>
</tr>
<tr>
<td>WFS_CIM_CITYPUNFIT</td>
<td>The cash in unit takes all unfit banknotes.</td>
</tr>
<tr>
<td>WFS_CIM_CITYPINDIVIDUAL</td>
<td>The cash in unit or recycler takes all types of banknotes specified in an individual list</td>
</tr>
</tbody>
</table>

**cUnitID**
The Cash Unit Identifier.

**cCurrencyID**
A three character array storing the ISO format Currency ID [see Ref. 2]. 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 wStatus field for this cash unit is WFS_CIM_STATCUNOVAL it is the responsibility of the application to assign a value to this field.

**ulValues**
Supplies the value of a single item in the cash unit. This value is expressed in minimum dispense units [see Section 4.2.5]. If the cCurrencyID field for this cash unit is empty then this field will contain 0. If the wStatus field for this cash unit is WFS_CIM_STATCUNOVAL it is the responsibility of the application to assign a value to this field.

**ulCashInCount**
Count of items that have entered the cash unit. This counter is incremented whenever a bill enters the physical cash unit for any reason. This value is persistent.

**ulCount**
Total number of notes of all types in the cash unit. If the cash unit is a recycle cash unit then this value may not be the same as the value of ulCashInCount. the value may be decremented as a result of a dispense operation on the CDM interface. For a retract cash unit this value specifies the number of retracts. This value is persistent.

**ulMaximum**
When the ulCount reaches this value the threshold event WFS_USRE_CIM_CASHUNITTHRESHOLD will be generated. If this value is non-0 then hardware sensors in the device do not trigger threshold events.
### usStatus

Describes the status of the cash unit as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_STATCUOK</td>
<td>The cash unit is in a good state.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUFULL</td>
<td>The cash in cash unit or recycle unit is full.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUHIGH</td>
<td>The cash in cash unit is almost full (threshold).</td>
</tr>
<tr>
<td>WFS_CIM_STATCUEMPTY</td>
<td>The recycle unit is empty.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUINOP</td>
<td>The cash in cash unit or recycle unit is inoperative.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUMISSING</td>
<td>The cash in cash unit is missing.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUNOVAL</td>
<td>The values of the specified cash unit are not available.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUNOREF</td>
<td>There is no reference value available for the notes in this cash unit. The cash unit has not been configured.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUMANIP</td>
<td>The cash unit has been changed when the device was not in the exchange state. Items cannot be accepted into this cash unit.</td>
</tr>
</tbody>
</table>

### bAppLock

This field does not apply to retract cash units. If this value is TRUE items cannot be accepted into the cash unit. This parameter is ignored if the hardware does not support this.

### lpNoteNumberList

Pointer to a WFSCIMNOTENUMBERLIST structure. If the cash unit is a retract cash unit this pointer will be NULL.

```c
typedef struct _wfs_cim_note_number_list
{
    USHORT usNumOfNoteNumbers;
    LPWFSCIMNOTENUMBER* lppNoteNumber;
} WFSCIMNOTENUMBERLIST, *LPWFSCIMNOTENUMBERLIST;
```

- **usNumOfNoteNumbers**
  
  Number of banknote types the cash unit contains, i.e. the size of the `lppNoteNumber` list.

- **lppNoteNumber**
  
  List of banknote numbers the cash unit contains. A pointer to an array of pointers to WFSCIMNOTENUMBER structures:

```c
typedef struct _wfs_cim_note_number
{
    USHORT usNoteID;
    ULONG ulCount;
} WFSCIMNOTENUMBER, *LPWFSCIMNOTENUMBER;
```

- **usNoteID**
  
  Identification of note type.

- **ulCount**
  
  Actual count of items. This value is persistent. The value is incremented each time items are moved to a cash unit by a `WFSEexecute` command. In the case of recycle cash units this count is decremented whenever items leave the cash unit.

### usNumPhysicalCUs

This value indicates the number of physical cash unit structures returned. It must be at least 1.

### lppPhysical

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

```c
typedef struct _wfs_cim_physicalcu
{
    LPSTR lpPhysicalPositionName;
    CHAR cUnitID[5];
    ULONG ulCashInCount;
    ULONG ulCount;
    ULONG ulMaximum;
} WFS_CIM_PHYSICALCU, *LPWFS_CIM_PHYSICALCU;
```
USHORT usPStatus;
BOOL bHardwareSensors;
LPSTR lpszExtra;
}

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

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

ulCashInCount
Count of items that have entered the cash in unit. This counter is incremented whenever a bill enters the physical cash unit for any reason. This value is persistent.

ulCount
Actual count of items in the physical cash unit. If the cash unit is a recycle cash unit then this value may not be the same as the value of ulCashInCount. This value is persistent.

ulMaximum
Maximum count of items in the physical cash unit. This is only for informational purposes. No threshold event will be generated.

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_CIM_STATCUOK</td>
<td>The cash unit is in a good state.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUFULL</td>
<td>The cash unit is full.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUHIGH</td>
<td>The cash unit is almost full (nearing the threshold defined by ulMaximum).</td>
</tr>
<tr>
<td>WFS_CIM_STATCULOW</td>
<td>The cash unit is almost empty (nearing the threshold defined by ulMinimum).</td>
</tr>
<tr>
<td>WFS_CIM_STATCUEMPTY</td>
<td>The cash unit is empty.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUNOP</td>
<td>The cash unit is inoperative.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUMISSING</td>
<td>The cash unit is missing.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUNOVAL</td>
<td>The values of the specified cash unit are not available.</td>
</tr>
<tr>
<td>WFS_CIM_STATCUNOREF</td>
<td>There is no reference value available for the notes in this cash unit. The cash unit has not been configured.</td>
</tr>
<tr>
<td>WFS_CIM_STATMANIP</td>
<td>The cash unit has been changed when the device was not in the exchange state.</td>
</tr>
</tbody>
</table>

bHardwareSensors
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.

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

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

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

Comments
None.
4.2.4. WFS_INF_CIM_TELLER_INFO

Description
This command allows the application to obtain counts for each currency assigned to the teller. It 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
LPWFSCIMTELLERINFO lpTellerInfo;
typedef struct _wfs_cim_teller_info
{
    USHORT usTellerID;
    CHAR cCurrencyID[3];
} WFSCIMTELLERINFO, *LPWFSCIMTELLERINFO;

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

cCurrencyID
Three character ISO format currency identifier [Ref. 2] This parameter can be an array of three ASCII 0x20h characters. In this case information on all currencies will be returned.

Output Param
LPWFSCIMTELLERDETAILS* lppTellerDetails;
typedef struct _wfs_cim_teller_details
{
    USHORT usTellerID;
    WORD fwInputPosition;
    WORD fwOutputPosition;
    LPWFSCIMTELLERTOTALS* lppTellerTotals;
} WFSCIMTELLERDETAILS, *LPWFSCIMTELLERDETAILS;

usTellerID
Identification of teller.

fwInputPosition
The input position assigned to the teller for cash entry. 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_CIM_POSNULL</td>
<td>No position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>The left position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>The right position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>The center position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>The top position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>The bottom position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRONT</td>
<td>The front position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>The 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 set to one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSNULL</td>
<td>No position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>The left position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>The right position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>The center position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>The top position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>The bottom position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>The front position is assigned to the Teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>The rear position is assigned to the Teller.</td>
</tr>
</tbody>
</table>
CWA 14050-19:2000

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

```c
typedef struct _wfs_cim_teller_totals {  
    USHORT    usTellerID;
    char       cCurrencyID[3];
    ULONG      ulBills;
    ULONG      ulCoins;
    ULONG      ulCashIn;
    ULONG      ulCashBox;
    ULONG      ulItemsReceived;
    ULONG      ulItemsDispensed;
    ULONG      ulCoinsReceived;
    ULONG      ulCoinsDispensed;
    ULONG      ulCashBoxReceived;
    ULONG      ulCashBoxDispensed;
} WFSCIMTELLERTOTALS, * LPWFSCIMTELLERTOTALS
```

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

**ulItemsReceived**  
The total amount of item currency (excluding coins) accepted. The amount is expressed in minimum dispense units (see WFS_INF_CIM_CURRENCY_EXP).

**ulItemsDispensed**  
The total amount of item currency (excluding coins) accepted. The amount is expressed in minimum dispense units (see WFS_INF_CIM_CURRENCY_EXP).

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

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

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

**ulCashBoxDispensed**  
The total amount of cash box currency dispensed. The amount is expressed in minimum dispense units (see WFS_INF_CIM_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_CIM_INVALIDCURRENCY</td>
<td>Specified currency not currently available</td>
</tr>
<tr>
<td>WFS_ERR_CIM_INVALIDTELLERID</td>
<td>Invalid Teller ID</td>
</tr>
</tbody>
</table>

**Comments**  
None.

**4.2.5. WFS_INF_CIM_CURRENCY_EXP**

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

**Input Param**  
None.

**Output Param**  
LPWFSINF_CURRENCY_EXP * lppCurrencyExp;  
Pointer to a null-terminated array of pointers to currency exponent structures:

```c
typedef struct _wfs_cim_currency_exp {  
    CHAR       cCurrencyID[3];
} WFSINF_CURRENCY_EXP, * LPWFSINF_CURRENCY_EXP;
```
SHORT sExponent;
)
WFSCIMCURRENCYEXP, *LPWFSCIMCURRENCYEXP;

\textit{cCurrencyID}
Currency identifier in ISO 4217 format [see Ref. 2].

\textit{sExponent}
Currency exponent in ISO 4217 format [see Ref. 2].

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

\section*{4.3. New Execute Commands}

\subsection*{4.3.1. WFS_CMD_CIM_RESET}

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

Persistent values, such as counts and configuration information are not cleared by this command.

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

If items are found inside the device the WFS_SRVE_CIM_MEDIADETECTED event will be
generated to inform the application where the items have actually been moved to.

\textbf{Input Param}
\begin{verbatim}
LPWFSCIMITEMPOSITION lpResetIn;
typedef struct _wfs_cim_itemposition 
{
    USHORT usNumber;
    LPWFSCIMRETRACT lpRetractArea;
    WORD fwOutputPosition;
} WFSCIMITEMPOSITION * LPWFSCIMITEMPOSITION;
\end{verbatim}

\textit{usNumber}
The \textit{usNumber} of the cash unit to which items which were inside the CIM when the reset was
issued should be moved. If the items should be moved to an output position this value is 0.

\textit{lpRetractArea}
This field is only used if the cash unit specified by \textit{usNumber} is a retract cash unit. In all other
cases this field is set to 0. For a description of this structure see the WFSCIMRETRACT structure
defined in Error! Reference source not found..

\textit{fwOutputPosition}
The output position to which items are to be moved. If the \textit{usNumber} is non-zero then this field
will be 0. The value is set to one of the following values:

\begin{tabular}{ll}
\hline
Value & Meaning \\
\hline
WFS_CIM_POSNULL & Take the default configuration. \\
WFS_CIM_POSOUTLEFT & Move items to the left output position. \\
WFS_CIM_POSOUTRIGHT & Move items to the right output position. \\
WFS_CIM_POSOUTCENTER & Move items to the center output position. \\
WFS_CIM_POSOUTTOP & Move items to the top output position. \\
WFS_CIM_POSOUTBOTTOM & Move items to the bottom output position. \\
WFS_CIM_POSOUTFRONT & Move items to the front output position. \\
WFS_CIM_POSOUTREAR & Move items to the rear output position. \\
\hline
\end{tabular}

If the application does not wish to specify a cash unit or position it can set this value to NULL. In
this case the service provider will determine where to move any items found.
Output Param  None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_CASHUNITERROR</td>
<td>A cash unit caused an error. A</td>
</tr>
<tr>
<td>WFS_ERR_CIM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_INVALIDCASHUNIT</td>
<td>The cash unit number specified is not valid.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in the exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_CASHINACTIVE</td>
<td>A Cash-In transaction is 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_USRE_CIM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_CASUNITERROR</td>
<td>A cash unit caused an error.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_MEDIADETECTED</td>
<td>Media was detected during the reset.</td>
</tr>
</tbody>
</table>

Comments  None.

4.3.2. WFS_CMD_CIM_CONFIGURE_CASH_IN_UNITS

Description  This command is used to alter the banknote types a cash in unit or recycle unit can take. The cash units which are affected by this command must be empty.

The values set by this command are persistent.

Input Param  LPWFS_CIM_CASHINTYPE * lppCashInType;

Pointer to a NULL terminated array of pointers to cash in type structures. Only the cash units which are to be configured should be sent in this parameter:

```c
typedef struct _wfs_cim_cash_in_type
{
    USHORT usNumber;
    DWORD dwType;
    LPUSHORT lpusNoteIDs;
} WFS_CIM_CASHINTYPE, * LPWFS_CIM_CASHINTYPE;
```

usNumber  Logical number of the cash unit.

dwType  Type of cash in unit or recycle unit. Specified as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_CITYPALL</td>
<td>The cash in unit accepts all banknote types.</td>
</tr>
<tr>
<td>WFS_CIM_CITYPUNFIT</td>
<td>The cash in unit accepts all unfit banknotes.</td>
</tr>
<tr>
<td>WFS_CIM_CITYPINDIVIDUAL</td>
<td>The cash in unit or recycle unit accepts all types of bank notes specified in the following list.</td>
</tr>
</tbody>
</table>

lpusNoteIDs  Pointer to a NULL terminated list of unsigned shorts which contains the note IDs of the bank notes the cash in cash unit or recycle unit can take.

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_CIM_INVALIDCASHUNIT</td>
<td>Invalid cash unit ID. This error will also be created if an invalid logical number of a cash unit is given.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM service 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_CIM_CASHUNITINFOCHANGED</td>
<td>A cash unit was changed.</td>
</tr>
</tbody>
</table>

### Comments

None.

4.3.3. **WFS_CMD_CIM_CONFIGURE_NOTETYPES**

**Description**

This command is used to configure the note types the banknote reader will recognise during cash in. All note types the banknote reader has to recognise must be given in the input structure. If an unknown note type is given the error code WFS_ERR_UNSUPPORTED_DATA will be returned. The values set by this command are persistent.

**Input Param**

- **LPUSHORT lpusNoteIDs**

  *lpusNoteIDs*
  
  Pointer to a NULL terminated list of unsigned shorts which contains the note IDs of the bank notes the banknote reader can accept.

**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_CIM_EXCHANGEACTIVE</td>
<td>The CIM 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.

4.4. **Changes To Execute Commands which previously existed in the CDM**

4.4.1. **WFS_CMD_CIM_RETRACT**

**Description**

This command retracts items from an output position. Retracted items will be moved to either a retract bin, the transport or an intermediate stacker area. After the items are retracted the shutter is closed automatically.

**Input Param**

- **LPWFS_CIM_RETRACT lpRetract**

  
  *lpRetract*
  
  struct _wfs_cim_retract
  
  ```
  { WORD fwOutputPosition;
  USHORT usRetractArea;
  USHORT usIndex;
  } WFS_CIM_RETRACT, *LPWFS_CIM_RETRACT;
  ```

  Pointer to a NULL terminated list of unsigned shorts which contains the note IDs of the bank notes the banknote reader can accept.
fwOutputPosition
Specifies the output position from which to retract the bills. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSNULL</td>
<td>The default configuration information should be used.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Retract items from the left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Retract items from the right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Retract items from the center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Retract items from the top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Retract items from the bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Retract items from the front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</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_CIM_RA_RETRACT</td>
<td>Retract the items to a retract cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_RA_TRANSPORT</td>
<td>Retract the items to the transport.</td>
</tr>
<tr>
<td>WFS_CIM_RA_STACKER</td>
<td>Retract the items to the intermediate stacker area.</td>
</tr>
<tr>
<td>WFS_CIM_RA_BILLCASSETTES</td>
<td>Retract the items to the recycle cash units.</td>
</tr>
</tbody>
</table>

usIndex
If usRetractArea is set to WFS_CIM_RA_RETRACT this field is the logical retract position inside the container into which the cash is to be retracted. This logical number starts with a value of one (1) for the first retract position and increments by one for each subsequent position. If the container contains several logical retract cash units (of type WFS_CIM_TYPERETRACTCASSETTE in command WFS_INF_CIM_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 WFSCIMCASHINFO. The maximum value of usIndex is the sum of the ulMaximum of each retract cash unit. If usRetractArea is not set to WFS_CIM_RA_RETRACT the value of this field is ignored.

Output Param None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_CASHUNITERROR</td>
<td>The retract bin caused a problem. A WFS_EXECUTE_EVENT with an id of WFS_EXEE_CIM_CASHUNITERROR will be posted with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOBILLS</td>
<td>There were no presented bills to retract.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_NOITEMS</td>
<td>There were no items to retract.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SHUTTERNOTCLOSED</td>
<td>The shutter failed to close.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_BILLSTAKEN</td>
<td>Bills were present at the exit at the start of the operation, but were removed before the operation was complete, so some or all of the bills were not retracted.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_ITEMSTAKEN</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDRETRACT</td>
<td>Retract function is invalid for this system.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_INVALIDRETRACTPOSITION</td>
<td>The usIndex is invalid for this system.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_NOTRETRACTAREA</td>
<td>The retract area specified in usRetractArea is not supported.</td>
</tr>
</tbody>
</table>
In addition to the generic events defined in [Ref. 1], the following additional 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_CIM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in the retract bin.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>An error occurred while attempting to retract to the retract bin.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_NOTEERROR</td>
<td>A note detection error occurred.</td>
</tr>
</tbody>
</table>

None.

4.4.2. WFS_CMD_CIM_CASH_IN

This command moves items into the CIM from an input position. The items may pass through the banknote reader for identification. Failure to identify items does not mean that the command has failed - even if some or all of the items are rejected by the banknote reader, the command may return WFS_SUCCESS. In this case a WFS_EXEE_CIM_INPUTREFUSE event will be sent to report the rejection.

If the device does not have a banknote reader then the output parameter will be NULL.

If the device has a cash-in stacker then this command will cause inserted items to be moved there. Items will be held on the stacker until the current Cash-In Transaction is either cancelled by WFS_CMD_CIM_ROLLBACK or confirmed by WFS_CMD_CIM_CASH_IN_END. If there is no cash-in stacker then this command will move items directly to the cash units and WFS_CMD_CIM_ROLLBACK will not be supported.

The bShutterControl field of the LPWFSCIMCAPS structure returned from the WFS_INF_CIM_CAPABILITIES query will determine whether the shutter is controlled implicitly by this command or whether the application must explicitly open and close the shutter using the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands.

It is possible that a device may divide bill or coin accepting into a series of sub-operations under hardware control. In this case a WFS_EXEE_CIM_SUBCASHIN event may be sent after each sub-operation, if the hardware capabilities allow it.

Input Param

None.

Output Param

LPWFSCIMNOTENUMBERLIST lpNoteNumberList;

List of banknote numbers which have been identified and accepted during execution of this command. If the whole input was refused then this parameter will be NULL and the WFS_EXEE_CIM_INPUTREFUSE event will be generated. If only part of the input was refused then this parameter will contain the banknote numbers of the accepted items and the WFS_EXEE_CIM_INPUTREFUSE event will be generated. For a description of the LPWFSCIMNOTENUMBERLIST structure see the WFS_INF_CIM_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_CIM_CASHUNITERROR</td>
<td>A problem occurred with a Cash Unit. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDCURRENCY</td>
<td>Specified currency not currently available</td>
</tr>
<tr>
<td>WFS_ERR_CDM_INVALIDTELLERID</td>
<td>Teller ID not present in service provider’s teller ID list</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOCASHINSTARTED</td>
<td>The WFS_CMD_CDM_CASH_IN_START was not issued before</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_CIM_CASHUNITERROR</td>
<td>A problem occurred with a Cash Unit.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INPUTREFUSE</td>
<td>A part or all of the amount of the cash in order was refused.</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>WFS_EXEE_CIM_NOTEERROR</td>
<td>A note detection error occurred.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_SUBCASHIN</td>
<td>A Cash In sub-operation has completed. If the Cash In operation has been divided up into a</td>
</tr>
<tr>
<td></td>
<td>series of sub-operations under hardware control this event is generated each time one of</td>
</tr>
<tr>
<td></td>
<td>the sub-cash-in operations completes successfully. It may be used for progress reporting.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSINSERTED</td>
<td>Items have been inserted into the cash in position by the user.</td>
</tr>
</tbody>
</table>

**Comments**

None.

### 4.4.3. WFS_CMD_CIM_OPEN_SHUTTER

**Description**

This command opens the shutter.

**Input Param**

LPWORD lpwPosition;

*lpwPosition*

Specifies which shutter is to be opened. If the application does not need to specify the shutter, this field can be set to NULL or to WFS_CIM_POSNULL. Otherwise 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_CIM_POSNULL</td>
<td>The default configuration information should be used.</td>
</tr>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>Open the shutter of the left input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>Open the shutter of the right input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>Open the shutter of the center input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>Open the shutter of the top input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>Open the shutter of the bottom input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRONT</td>
<td>Open the shutter of the front input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>Open the shutter of the rear input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Open the shutter of the left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTHRIGHT</td>
<td>Open the shutter of the right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTHCENTER</td>
<td>Open the shutter of the center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Open the shutter of the top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Open the shutter of the bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Open the shutter of the front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Open the shutter of 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:
### 4.4.3.1. WFS_CMD_CIM_CLOSE_SHUTTER

**Description**
This command closes the shutter.

**Input Param**
LPWORD lpfwPosition;

*lpfwPosition*
Specifies which shutter to be closed. If the application does not need to specify the shutter, this field can be set to NULL or to WFS_CIM_POSNULL. Otherwise 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_CIM_POSNULL</td>
<td>The default configuration information should be used.</td>
</tr>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>Close the shutter of the left input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>Close the shutter of the right input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>Close the shutter of the center input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>Close the shutter of the top input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>Close the shutter of the bottom input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRONT</td>
<td>Close the shutter of the front input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>Close the shutter of the rear input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Close the shutter of the left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Close the shutter of the right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Close the shutter of the center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Close the shutter of the top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Close the shutter of the bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Close the shutter of the front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Close the shutter of 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_CIM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SHUTTERCLOSED</td>
<td>Shutter was already closed.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM service is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SHUTTERNOTCLOSED</td>
<td>Shutter failed to close.</td>
</tr>
</tbody>
</table>

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

**Comments**
None.

### 4.4.4. WFS_CMD_CIM_SET_TELLER_INFO

**Description**
This command allows the application to initialize counts for each currency assigned to the teller. The values set by this command are persistent. This command only applies to Teller CIMs.
Input Param

```c
typedef struct _wfs_cim_teller_update {
    USHORT  usAction;
    LPWFSCIMTELLERINFO  lpTellerDetails;
} WFSCIMTELLERUPDATE *LPWFSCIMTELLERUPDATE;
```

**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_CIM_CREATE_TELLER</td>
<td>A Teller is to be added.</td>
</tr>
<tr>
<td>WFS_CIM_MODIFY_TELLER</td>
<td>Information about an existing Teller is to be modified.</td>
</tr>
<tr>
<td>WFS_CIM_DELETE_TELLER</td>
<td>A teller is to be removed.</td>
</tr>
</tbody>
</table>

**lpTellerDetails**
For a specification of the struct WFSCIMTELLERINFO please refer to the WFS_INF_CIM_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_CIM_INVALIDCURRENCY</td>
<td>The specified currency is not currently available.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_INVALIDTELLERID</td>
<td>The Teller ID is invalid.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_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 additional 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_CIM_TELLERINFOCHANGED</td>
<td>Teller information has been created, modified or deleted.</td>
</tr>
</tbody>
</table>

Comments
None.

### 4.4.5. WFS_CMD_CIM_SET_CASH_UNIT_INFO

**Description**
This command is used to adjust information about the status and contents of the cash units present in the CIM.

This command generates the service event WFS_SRVE_CIM_CASHUNITINFOCHANGED to inform applications that cash unit information 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 WFSCIMCASHIN structure may be updated by this command:

- `ulCount`
- `ulCashInCount`
- `ulMaximum`
- `bAppLock`

As may the following fields of the WFSCIMPHCU structure:

- `ulCashInCount`
- `ulCount`

Any other changes must be performed via an exchange operation.
If the fields ulCount and ulCashInCount of lppPhysical are set to 0 by this command, the application is indicating that it does not wish counts to be maintained for the physical cash units. Counts on the logical cash units will still be maintained and can be used by the application. If the physical counts are set by this command then the logical count will be the sum of the physical counts and any value sent as a logical count will be ignored.

**Input Param**

LPWFS_CIMCAS_H_INFO lpCUInfo;
The LPWFS_CIMCAS_H_INFO structure is specified in the documentation of the WFS_INF_CIM_CASH_UNIT_INFO command. All cash units must be sent, not just the cash units whose values are to be changed.

**Output Param**

None.

**Error Codes**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CDM_CASHUNITERROR</td>
<td>A cash unit specified caused a problem. A WFS_EXEE_CDM_CASHUNITERROR execute event is posted with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_INVALIDTELLERID</td>
<td>Invalid Teller ID.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_INVALIDCASHUNIT</td>
<td>Invalid cash unit ID.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in an exchange state.</td>
</tr>
</tbody>
</table>

**Events**

In addition to the generic events defined in [Ref. 1], the following additional 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_CIM_CASHUNITITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_CASHUNITINFOCHANGED</td>
<td>A cash unit was updated as a result of this command.</td>
</tr>
</tbody>
</table>

**Comments**

None.

### 4.4.6. WFS_CMD_CIM_START_EXCHANGE

**Description**

This command puts the CIM 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_CIM_SET_CASH_UNIT_INFO command all changes to a cash unit must take place while the cash unit is in an exchange state.

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

The command returns current cash unit information in the form described in the documentation of the WFS_INF_CIM_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 obtained exclusive control of the CIM.

This command may return WFS_SUCCESS even if WFS_EXEE_CIM_CASHUNITERROR events are generated. If this command returns WFS_SUCCESS or WFS_ERR_CIM_EXCHANGE_ACTIVE the CIM is in an exchange state.

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

- WFS_CMD_CIM_END_EXCHANGE
- Any WFS[Async]GetInfo commands
- WFS_Close - This will end the exchange state.

Any other commands will result in the error WFS_ERR_CIM_EXCHANGEACTIVE being generated.
If an error is returned by this command, the `WFS_CMD_CIM_CASH_UNIT_INFO` command should be used to determine the cash unit information.

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

**CDM**

(Lock)

`WFS_CMD_CDM_START_EXCHANGE`

...exchange action...

`WFS_CMD_CDM_END_EXCHANGE`

(Unlock)

**CIM**

(Lock)

`WFS_CMD_CIM_START_EXCHANGE`

...exchange action...

`WFS_CMD_CIM_END_EXCHANGE`

(Unlock)

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

**Input Param**

`LPWFS_CIM_START_EX lpStartEx;`

define struct _wfs_cim_start_ex {
    WORD fwExchangeType;
    USHORT usTellerID;
    USHORT usCount;
    LPUSHORT lpusCUNumList;
    LPWFS_CIM_OUTPUT lpOutput;
} WFS_CIM_START_EX, *LPWFS_CIM_START_EX;

`fwExchangeType`

Specifies the type of the 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><code>WFS_CIM_EXBYHAND</code></td>
<td>The cash units will be replenished manually either by filling or emptying the cash unit by hand or by replacing the cash unit.</td>
</tr>
<tr>
<td><code>WFS_CIM_EXTOCASSETTES</code></td>
<td>Items will be moved from the replenishment container to the bill cash units.</td>
</tr>
<tr>
<td><code>WFS_CDM_EXDEPOSITINTO</code></td>
<td>Replenish of the bill cassettes from the deposit entrance.</td>
</tr>
<tr>
<td><code>WFS_CDM_EXCOLLECTALL</code></td>
<td>Empty by moving bills from the bill cassettes to a bill output position.</td>
</tr>
<tr>
<td><code>WFS_CDM_EXCOLLECTFROMREPCASHUNIT</code></td>
<td>Empty the replenishment container to a bill output position.</td>
</tr>
<tr>
<td><code>WFS_CIM_CLEARRECYCLER</code></td>
<td>Items will be moved from a recycle cash unit to a cash unit or output position.</td>
</tr>
<tr>
<td><code>WFS_CIM_DEPOSITINTO</code></td>
<td>Items will be moved from the deposit entrance to the bill cash units.</td>
</tr>
</tbody>
</table>

`usTellerID`

Identification of teller. **If the device is a Self-Service CIM 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.
PipeConNumList
Pointers to an array of unsigned shorts containing the logical numbers of the cash units to be exchanged.

lpOutput
This parameter is used when the exchange type is WFS_CIM_CLEARRECYCLER, i.e. a recycle cash unit is to be emptied.

typedef struct _wfs_cim_output {
    USHORT usLogicalNumber;
    WORD fwPosition;
    USHORT usNumber;
} WFSCIMOUTPUT, * LPWFSCIMOUTPUT;

usLogicalNumber
Logical number of recycle unit to be emptied.

fwPosition
Determines to which position the cash should be moved as a combination of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSNULL</td>
<td>Move items to a cash unit. If no cash unit is specified in usNumber, use the default output position</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Move items to the left output position</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Move items to the right output position</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Move items to the center output position</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Move items to the top output position</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Move items to the bottom output position</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Move items to the front output position</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Move items to the rear output position</td>
</tr>
</tbody>
</table>

usNumber
Logical number of the cash unit the items are to be moved to.

Output Param LPWFSCIMCASHININFO lpCUInfo;
The LPWFSCIMCASHININFO structure is specified in the documentation of the WFS_INF_CIM_CASH_UNIT_INFO command. Information on all the CIM cash units will be returned.

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_CIM_INVALIDTELLERID</td>
<td>Invalid Teller ID. This error will never be generated by a Self-Service CIM.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_CASHUNITERROR</td>
<td>An error occurred with a cash unit while performing the exchange operation. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_TOOMANYITEMS</td>
<td>This error is generated if the contents of the recycler cash unit can not be completely emptied to the output position. The maximum possible number of items is moved to the output position. The CIM is already in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>An error occurred while performing the exchange operation.</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_CIM_CASHUNITERROR</td>
<td>An error occurred while performing the exchange operation.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_NOTERROR</td>
<td>A notes detection error has occurred.</td>
</tr>
</tbody>
</table>

Comments
None.
4.4.7. WFS_CMD_CIM_END_EXCHANGE

**Description**

This command will end the exchange state. If any physical action took place as a result of the `WFS_CMD_CIM_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_CIM_CASH_UNIT_INFO` command.

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

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

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

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

**Input Param**

_`LPWFCIMCASHINFO` lpCUInfo;

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

**Output Param**

None.

**Error Codes**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>WFS_ERR_CIM_INVALIDTELLERID</code></td>
<td>Invalid Teller ID</td>
</tr>
<tr>
<td><code>WFS_ERR_CIM_CASHUNITERROR</code></td>
<td>This error is returned if there is a problem with the values set for a cash unit. A <code>WFS_EXEE_CIM_CASHUNITERROR</code> event will be sent with the details.</td>
</tr>
<tr>
<td><code>WFS_ERR_CIM_NOEXCHANGEACTIVE</code></td>
<td>There is no exchange active</td>
</tr>
<tr>
<td><code>WFS_ERR_CIM_INVALIDCASHUNIT</code></td>
<td>Invalid cash unit ID</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><code>WFS_USRE_CIM_CASHUNITITHTHRESHOLD</code></td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
<tr>
<td><code>WFS_SRVE_CIM_CASHUNITINFOCHANGED</code></td>
<td>A cash unit was changed.</td>
</tr>
<tr>
<td><code>WFS_EXEE_CIM_CASHUNITERROR</code></td>
<td>The values of the cash unit structures are incorrect. The cash unit structure that is incorrect is returned as a parameter on this event.</td>
</tr>
</tbody>
</table>

**Comments**

None.

4.4.8. WFS_CMD_CIM_OPEN_SAFE_DOOR

**Description**

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

**Input Param**

None.

**Output Param**

None.
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_CIM_EXCHANGEACTIVE</td>
<td>The CIM 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.

4.4.9. WFS_CMD_CIM_CASH_IN_START

Description

Before initiating a Cash-In operation, an application must issue the WFS_CMD_CIM_CASH_IN_START command to begin a Cash-In Transaction. During a Cash-In Transaction any number of WFS_CMD_CIM_CASH_IN commands may be issued. The transaction is ended when either a WFS_CMD_CIM_ROLLBACK or WFS_CMD_CIM_CASH_IN_END command is sent.

Input Param

LPWFS_CIM_CASH_IN_START lpCashInStart;

typedef struct _wfs_cim_cash_in_start
{
    USHORT usTellerID;
    BOOL bUseRecycleUnits;
    WORD fwOutputPosition;
    WORD fwInputPosition;
} WFSCIM_CASH_IN_START, *LPWFSCIM_CASH_IN_START;

usTellerID

Identification of teller. This field is not applicable to Self-Service CIMs and should be set to 0.

bUseRecycleUnits

Specifies whether or not the recycle cash units should be used for money cashed in during the transaction period. This parameter will be ignored if there are no recycle cash units or the hardware does not support this.

fwOutputPosition

The output position where the items will be presented to the customer in the case of a cash in rollback. The position is set to one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSNULL</td>
<td>Default configuration.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Top output position</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Bottom output position</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Front output position</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Rear output position</td>
</tr>
</tbody>
</table>

fwInputPosition

Specifies from which position the cash should be inserted. The position is set to one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSNULL</td>
<td>Default configuration.</td>
</tr>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>Left input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>Right input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>Center input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>Top input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>Bottom input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRT</td>
<td>Front input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>Rear input position.</td>
</tr>
</tbody>
</table>

Output Param

None.
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_CIM_INVALIDTELLERID</td>
<td>The Teller ID is invalid.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in the exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_CASHINACTIVE</td>
<td>The CIM is already in the cash in state due to a previous WFS_CMD_CIM_CASH_IN_START command.</td>
</tr>
</tbody>
</table>

**Events**

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

**Comments**

None.

### 4.4.10. WFS_CMD_CIM_CASH_IN_END

**Description**

This command ends a Cash-In Transaction. If items are on the stacker as a result of a WFS_CMD_CIM_CASH_IN command, these items are moved into the cash-in cash units or the recycle units.

The Cash-In transaction is ended even if this command does not complete successfully.

**Input Param**

None.

**Output Param**

<table>
<thead>
<tr>
<th>lpCashInfo</th>
</tr>
</thead>
<tbody>
<tr>
<td>List of cash units that have taken banknotes or coins and the type of banknotes or coins they have taken. For a description of the WFSCIMCASHINFO structure see the definition of the WFS_INF_CIM_CASH_UNIT_INFO command.</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_CIM_CASHUNITERROR</td>
<td>A problem occurred with a Cash Unit. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_NOITEMS</td>
<td>There were no items to cash in.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_NOCASHINACTIVE</td>
<td>There is no Cash-In transaction active.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_POSITION_NOT_EMPTY</td>
<td>The input or output position is not empty. The WFS_CMD_CDM_CASH_IN_START was not issued before</td>
</tr>
<tr>
<td>WFS_ERR_CDM_NOCASHINSTARTED</td>
<td></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_CIM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has occurred in one of the cash units.</td>
</tr>
<tr>
<td>WFS_SRV_E_CIM_CASHUNITINFOCHANGED</td>
<td>A cash unit was changed.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>A problem occurred with the cash unit.</td>
</tr>
</tbody>
</table>

**Comments**

None.

### 4.4.11. WFS_CMD_CIM_CASH_IN_ROLLBACK

**Description**

A Cash-In operation has to be handled as a transaction that can be rolled back if a difference occurs between the amount counted by the CIM and the amount inserted. This command is used
to roll back a Cash-In transaction. It causes all the notes cashed in since the last WFS_CMD_CIM_CASH_IN_START command to be returned to the customer.

This command ends the current Cash-In Transaction. The Cash-In transaction is ended even if this command does not complete successfully.

The bShutterControl field of the LPWFSCIMCAPS structure returned from the WFS_INF_CIM_CAPABILITIES query will determine whether the shutter is controlled implicitly by this command or whether the application must explicitly control the shutter using the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands.

**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_NOCASHINSTARTED</td>
<td>The WFS_CMD_CDM_CASH_IN_START was not issued before</td>
</tr>
<tr>
<td>WFS_ERR_CIM_CASHUNITERROR</td>
<td>A problem occurred with a Cash Unit. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SHUTTERNOTOPEN</td>
<td>Shutter failed to open.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in the exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_NOCASHINACTIVE</td>
<td>There is no current Cash-In Transaction.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_POSITION_NOT_EMPTY</td>
<td>The input or output position is not empty.</td>
</tr>
</tbody>
</table>

**Events**

In addition to the generic events defined in [Ref. 1], the following additional 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_CIM_CASHUNITERROR</td>
<td>A problem occurred with a Cash Unit.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>Either the items are available to the user or have been removed by the user, depending on the capability of the CIM.</td>
</tr>
</tbody>
</table>

**Comments**

None.

### 4.4.12. WFS_CMD_CIM_RESET

**Description**

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

Persistent values, such as counts and configuration information are not cleared by this command.

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

If items are found inside the device the WFS_SRVE_CIM_MEDIADETECTED event will be generated to inform the application where the items have actually been moved to.

**Input Param**

```c
typedef struct _wfs_cim_itemposition
{
    USHORT usNumber;
    LPWFSCIMRETRACT lpRetractArea;
    WORD fwOutputPosition;
} WFSCIMITEMPOSITION * LPWFSCIMITEMPOSITION;
```
The `usNumber` of the cash unit to which items which were inside the CIM when the reset was issued should be moved. If the items should be moved to an output position this value is 0.

This field is only used if the cash unit specified by `usNumber` is a retract cash unit. In all other cases this field is set to 0. For a description of this structure see the WFSCIMRETRACT structure defined in Error! Reference source not found..

The output position to which items are to be moved. If the `usNumber` is non-zero then this field will be 0. 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_CIM_POSNULL</td>
<td>Move items to the default configuration.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Move items to the left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Move items to the right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Move items to the center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Move items to the top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Move items to the bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Move items to the front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Move items to the rear output position.</td>
</tr>
</tbody>
</table>

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

**Output Param**
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_CASHUNITERROR</td>
<td>A cash unit caused an error. A</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>Event will be sent with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_INVALIDCASHUNIT</td>
<td>The cash unit number specified is not valid.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in the exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_CASHINACTIVE</td>
<td>A Cash-In transaction is 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_USRE_CIM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>A cash unit caused an error.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM.MEDIADETECTED</td>
<td>Media was detected during the reset.</td>
</tr>
</tbody>
</table>

**Comments**
None.

### 4.5. New Events

#### 4.5.1. WFS_SRVE_CIM_COUNTS_CHANGED

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

**Event Param**
```
LPWFSCIMCOUNTSCHANGED lpCountsChanged;
typedef struct _wfs_cim_counts_changed
{    USHORT usCount;
```
USHORT *lpusCUNumList;
} WFSCIMCOUNTSCHANGED, *LPWFSCIMCOUNTSCHANGED;

usCount
The size of lpusCUNumList.

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

Comments
None.

4.5.2. WFS_SRVE_CIM_ITEMSPRESENTED
Description
This service event specifies that items have been presented to the user and need to be taken.
Event Param
None.
Comments
None.

4.5.3. WFS_SRVE_CIM_ITEMSINSERTED
Description
This service event specifies that items have been inserted into the cash in position by the user.
Event Param
None.
Comments
None.

4.5.4. WFS_EXEE_CIM_NOTEERROR
Description
This execute event specifies the reason for a notes detection error during an operation which involves moving notes.
Event Param
USHORT lpusReason;

lpusReason
Specifies 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_CIM_DOUBLENOTEDETECTED</td>
<td>Double notes have been detected.</td>
</tr>
<tr>
<td>WFS_CIM_LONGNOTEDETECTED</td>
<td>A long note has been detected.</td>
</tr>
<tr>
<td>WFS_CIM_SKEWEDNOTE</td>
<td>A skewed note has been detected.</td>
</tr>
<tr>
<td>WFS_CIM_INCORRECTCOUNT</td>
<td>A bill counting error has occurred.</td>
</tr>
<tr>
<td>WFS_CIM_NOTESTOOCLOSE</td>
<td>Notes have been detected as being too close.</td>
</tr>
</tbody>
</table>

Comments
None.

4.5.5. WFS_EXEE_CIM_SUBCASHIN
Description
This execute event is generated when one of the sub-cash-in operations into which the cash in operation was divided has finished successfully.
Event Param
WFSCIMNOTENUMBERLIST lpNoteNumberList;

lpNoteNumberList
List of banknote numbers which have been identified and accepted during execution of the sub-cash-in. This parameter will contain the banknote numbers of the accepted items. For a
description of the LPWFSCIMNOTENUMBERLIST structure see the WFS_INF_CIM_CASH_UNIT_INFO command.

Comments None.

4.5.6. WFS_SRVE_CIM_MEDIADETECTED

Description This service event is generated when media is detected during a reset (WFS_CMD_CIM_RESET). The parameter on the event specifies the position of the media on completion of the reset. If the device has been unable to successfully move the items found then this parameter will be NULL.

Event Param LPWFSCIMITEMPOSITION lpPosition;

For a description of this parameter see WFS_CMD_CIM_RESET.

Comments None.

4.6. Changes to Events which previously existed in the CDM

4.6.1. WFS_SRVE_CIM_ITEMSTAKEN (former WFS_SRVE_CDM_BILLSTAKEN)

Description This service event specifies that items presented to the user have been taken.

Event Param None.

Comments None.

4.6.2. WFS_SRVE_CIM_CASHUNITINFOCHANGED

Description This service event specifies that a cash unit has changed in configuration. A physical cash unit may have been removed or inserted or a cash unit parameter may have changed. This event will also be posted on successful completion of the following commands:

WFS_CMD_CIM_SET_CASH_UNIT_INFO
WFS_CMD_CIM_END_EXCHANGE

Event Param LPWFSCIMCASHIN lpCashUnit;

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

Comments None.

4.6.3. WFS_SRVE_CIM_TELLERINFOCHANGED

Description This service event specifies that the counts assigned to the specified teller have been changed. This event is only returned as a result of a WFS_CMD_CIM_SET_TELLER_INFO command.

Event Param LPUSHORT lpusTellerID;

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

Comments None.
4.6.4. WFS_EXEE_CIM_CASHUNITERROR

**Description**
This execute event specifies that in a denominate or dispense command a cash unit was addressed which caused a problem.

**Event Param**

```c
LPWFSCIMCUERROR lpCashUnitError;
```

typedef struct _wfs_cim_cu_error
{
    WORD wFailure;
    LPWFSCIMCASHIN lpCashUnit;
} WFSCIMCUERROR, *LPWFSCIMCUERROR;

**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_CIM_CASHUNITEMPTY</td>
<td>Specified cash unit is empty.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITERROR</td>
<td>Specified cash unit has malfunctioned.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITFULL</td>
<td>Specified cash unit is full.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITITLOCKED</td>
<td>Specified cash unit is locked.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITINOTCONF</td>
<td>Specified cash unit is not configured due to being removed and/or replaced with a different cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITINVALID</td>
<td>Specified cash unit ID is invalid.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITINVALID</td>
<td>Attempt to change the settings of a self configuring cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_FEEDMODULEPROBLEM</td>
<td>A problem has been detected with the feeding module.</td>
</tr>
</tbody>
</table>

**lpCashUnit**
Pointer to the cash unit structure that caused the problem. For a description of the WFSCIMCASHIN structure see the definition of the WFS_INF_CIM_CASH_UNIT_INFO command.

**Comments**
None.

4.6.5. WFS_EXEE_CIM_INPUTREFUSE

**Description**
This execute event specifies that the device has refused either a portion or the entire amount of the cash in order.

**Event Param**

```c
LPUSHORT lpusReason;
```

**lpusReason**
Specifies the reason for refusing a part of the amount. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_CASHINUNITFULL</td>
<td>Cash unit is full.</td>
</tr>
<tr>
<td>WFS_CIM_INVALIDBILL</td>
<td>One or more of the items are invalid.</td>
</tr>
<tr>
<td>WFS_CIM_NOBILLSTODEPOSIT</td>
<td>There are no bills in the input area.</td>
</tr>
<tr>
<td>WFS_CIM_DEPOSITFAILURE</td>
<td>A deposit has failed for a reason other than one of the reasons above, and the failure is not a fatal hardware problem.</td>
</tr>
<tr>
<td>WFS_CIM_COMMINPCOMPFAILURE</td>
<td>Failure of a common input component which is shared by all cash units.</td>
</tr>
<tr>
<td>WFS_CIM_STACKERFULL</td>
<td>The intermediate stacker is full.</td>
</tr>
</tbody>
</table>

**Comments**
None.
5. Change to CDM C-Header file

### xfscdm.h

```c
#ifndef __INC_XFSCDM__H
#define __INC_XFSCDM__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

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

/* values of WFSCDMCAPS.wClass */
#define     WFS_SERVICE_CLASS_CDM               (3)
#define     WFS_SERVICE_CLASS_VERSION_CDM       0x0003
#define     WFS_SERVICE_CLASS_NAME_CDM          "CDM"

#define     CDM_SERVICE_OFFSET                  (WFS_SERVICE_CLASS_CDM * 100)

/* CDM Info Commands */
#define     WFS_INF_CDM_STATUS                  (CDM_SERVICE_OFFSET + 1)
#define     WFS_INF_CDM_CAPABILITIES            (CDM_SERVICE_OFFSET + 2)
#define     WFS_INF_CDM_CASH_UNIT_INFO          (CDM_SERVICE_OFFSET + 3)
#define     WFS_INF_CDM_TELLER_INFO             (CDM_SERVICE_OFFSET + 4)
#define     WFS_INF_CDM_TELLER_POSITIONS        (CDM_SERVICE_OFFSET + 5)
#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_CASH_IN                 (CDM_SERVICE_OFFSET + 6)
#define     WFS_CMD_CDM_OPEN_SHUTTER            (CDM_SERVICE_OFFSET + 7)
#define     WFS_CMD_CDM_CLOSE_SHUTTER           (CDM_SERVICE_OFFSET + 8)
#define     WFS_CMD_CDM_SET_TELLER_INFO         (CDM_SERVICE_OFFSET + 9)
#define     WFS_CMD_CDM_SET_CASH_UNIT_INFO      (CDM_SERVICE_OFFSET + 10)
#define     WFS_CMD_CDM_START_EXCHANGE          (CDM_SERVICE_OFFSET + 11)
#define     WFS_CMD_CDM_END_EXCHANGE            (CDM_SERVICE_OFFSET + 12)
#define     WFS_CMD_CDM_SET_MIX_TABLE           (CDM_SERVICE_OFFSET + 13)
#define     WFS_CMD_CDM_RESET                   (CDM_SERVICE_OFFSET + 14)
#define     WFS_CMD_CDM_CASH_UNIT_INFO          (CDM_SERVICE_OFFSET + 16)
#define     WFS_CMD_CDM_OPEN_SAFE_DOOR          (CDM_SERVICE_OFFSET + 17)
#define     WFS_CMD_CDM_SET_TELLER_POSITIONS    (CDM_SERVICE_OFFSET + 18)
#define     WFS_CMD_CDM_CASH_IN_START           (CDM_SERVICE_OFFSET + 19)
#define     WFS_CMD_CDM_CASH_IN_END             (CDM_SERVICE_OFFSET + 20)
#define     WFS_CMD_CDM_CASH_IN_ROLLBACK        (CDM_SERVICE_OFFSET + 21)
#define     WFS_CMD_CDM_RESET                   (CDM_SERVICE_OFFSET + 22)
#define     WFS_CMD_CDM_COUNT                   (CDM_SERVICE_OFFSET + 23)

/* CDM Messages */
#define     WFS_SRVE_CDM_SAFEDOOROPEN           (CDM_SERVICE_OFFSET + 1)
#define     WFS_SRVE_CDM_SAFEDOORCLOSED         (CDM_SERVICE_OFFSET + 2)
#define     WFS_USRE_CDM_CASHUNITTHRESHOLD      (CDM_SERVICE_OFFSET + 3)
#define     WFS_SRVE_CDM_CASHUNITINFOCHANGED    (CDM_SERVICE_OFFSET + 4)
```

```c
#}ndef __INC_XFSCDM__H
#define __INC_XFSCDM__H

#include <xfsapi.h>

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

/* CDM Inf Info Commands */
#define     WFS_SRVE_CDM_INFO                  (CDM_SERVICE_OFFSET + 1)
#define     WFS_SRVE_CDM_STATUS                (CDM_SERVICE_OFFSET + 2)
#define     WFS_SRVE_CDM_CASH_UNIT_INFO        (CDM_SERVICE_OFFSET + 3)
#define     WFS_SRVE_CDM_TELLER_INFO           (CDM_SERVICE_OFFSET + 4)
#define     WFS_SRVE_CDM_TELLER_POSITIONS      (CDM_SERVICE_OFFSET + 5)
#define     WFS_SRVE_CDM_CURRENCY_EXP          (CDM_SERVICE_OFFSET + 6)
#define     WFS_SRVE_CDM_MIX_TYPES             (CDM_SERVICE_OFFSET + 7)
#define     WFS_SRVE_CDM_MIX_TABLE             (CDM_SERVICE_OFFSET + 8)
#define     WFS_SRVE_CDM_PRESENT_STATUS        (CDM_SERVICE_OFFSET + 9)

/* CDM Execute Commands */
#define     WFS_SRVE_CDM_EXECUTE               (CDM_SERVICE_OFFSET + 1)
#define     WFS_SRVE_CDM_DENOMINATE            (CDM_SERVICE_OFFSET + 2)
#define     WFS_SRVE_CDM_DISPENSE              (CDM_SERVICE_OFFSET + 3)
#define     WFS_SRVE_CDM_PRESENT               (CDM_SERVICE_OFFSET + 4)
#define     WFS_SRVE_CDM_REJECT                (CDM_SERVICE_OFFSET + 5)
#define     WFS_SRVE_CDM_RETRACT               (CDM_SERVICE_OFFSET + 6)
#define     WFS_SRVE_CDM_CASH_IN               (CDM_SERVICE_OFFSET + 7)
#define     WFS_SRVE_CDM_OPEN_SHUTTER          (CDM_SERVICE_OFFSET + 8)
#define     WFS_SRVE_CDM_CLOSE_SHUTTER         (CDM_SERVICE_OFFSET + 9)
#define     WFS_SRVE_CDM_SET_TELLER_INFO       (CDM_SERVICE_OFFSET + 10)
#define     WFS_SRVE_CDM_SET_CASH_UNIT_INFO    (CDM_SERVICE_OFFSET + 11)
#define     WFS_SRVE_CDM_START_EXCHANGE        (CDM_SERVICE_OFFSET + 12)
#define     WFS_SRVE_CDM_END_EXCHANGE          (CDM_SERVICE_OFFSET + 13)
#define     WFS_SRVE_CDM_SET_MIX_TABLE         (CDM_SERVICE_OFFSET + 14)
#define     WFS_SRVE_CDM_RESET                 (CDM_SERVICE_OFFSET + 15)
#define     WFS_SRVE_CDM_CASH_UNIT_INFO        (CDM_SERVICE_OFFSET + 16)
#define     WFS_SRVE_CDM_OPEN_SAFE_DOOR        (CDM_SERVICE_OFFSET + 17)
#define     WFS_SRVE_CDM_SET_TELLER_POSITIONS  (CDM_SERVICE_OFFSET + 18)
#define     WFS_SRVE_CDM_CASH_IN_START         (CDM_SERVICE_OFFSET + 19)
#define     WFS_SRVE_CDM_CASH_IN_END           (CDM_SERVICE_OFFSET + 20)
#define     WFS_SRVE_CDM_CASH_IN_ROLLBACK      (CDM_SERVICE_OFFSET + 21)
#define     WFS_SRVE_CDM_RESET                 (CDM_SERVICE_OFFSET + 22)
#define     WFS_SRVE_CDM_COUNT                 (CDM_SERVICE_OFFSET + 23)
```
#define WFS_SRVE_CDM_TELLERINFOCHANGED (CDM_SERVICE_OFFSET + 5)
#define WFS_EXEE_CDM_DELAYEDDISPENSE (CDM_SERVICE_OFFSET + 6)
#define WFS_EXEE_CDM_STARTDISPENSE (CDM_SERVICE_OFFSET + 7)
#define WFS_EXEE_CDM_CASHUNITERROR (CDM_SERVICE_OFFSET + 8)
#define WFS_SRVE_CDM_ITEMSTAKEN (CDM_SERVICE_OFFSET + 9)
#define WFS_EXEE_CDM_PARTIALDISPENSE (CDM_SERVICE_OFFSET + 10)
#define WFS_EXEE_CDM_SUBDISPENSEOK (CDM_SERVICE_OFFSET + 11)
#define WFS_SRVE_CDM_ITEMSPRESENTED (CDM_SERVICE_OFFSET + 13)
#define WFS_EXEE_CDM_COUNTS_CHANGED (CDM_SERVICE_OFFSET + 14)
#define WFS_EXEE_CDM_INCOMPLETEDISPENSE (CDM_SERVICE_OFFSET + 15)
#define WFS_EXEE_CDM_NOTEERROR (CDM_SERVICE_OFFSET + 16)
#define WFS_EXEE_CDM_MEDIADETECTED (CDM_SERVICE_OFFSET + 17)

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

/* values of WFSCDMSTATUS.fwSafeDoor */
#define WFS_CDM_DOORNOTSUPPORTED (1)
#define WFS_CDM_DOOROPEN (2)
#define WFS_CDM_DOORCLOSED (3)
#define WFS_CDM_DOORUNKNOWN (5)

/* values of WFSCDMSTATUS.fwDispenser */
#define WFS_CDM_DISPOK (0)
#define WFS_CDM_DISPCUSTATE (1)
#define WFS_CDM_DISPCUSTOP (2)
#define WFS_CDM_DISPCUUNKNOWN (3)

/* values of WFSCDMSTATUS.fwIntermediateStacker */
#define WFS_CDM_ISEMPTY (0)
#define WFS_CDM_ISNOTEMPTY (1)
#define WFS_CDM_ISNOTEMPTYCUST (2)
#define WFS_CDM_ISNOTEMPTYUNK (3)
#define WFS_CDM_ISUNKNOWN (4)
#define WFS_CDM_ISNOTSUPPORTED (5)

/* values of WFSCDMOUTPOS.fwShutter */
#define WFS_CDM_SHTCLOSED (0)
#define WFS_CDM_SHTOOPEN (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)
```c
#define WFS_CDM_TPSTATNOTSUPPORTED (4)

/* values of WFSCDMCAPS.fwType */
#define WFS_CDM_TELLERBILL (0)
#define WFS_CDM_SELFSERVICEBILL (1)
#define WFS_CDM_TELLERCOIN (2)
#define WFS_CDM_SELFSERVICECOIN (3)

/* values of WFSCDMCAPS.fwRetractAreas */
/* values of WFSCDMRETRACT.usRetractArea */
#define WFS_CDM_RA_RETRACT (0x0001)
#define WFS_CDM_RA_TRANSPORT (0x0002)
#define WFS_CDM_RA_STACKER (0x0004)
#define WFS_CDM_RA_REJECT (0x0008)
#define WFS_CDM_RA_NOTSUPP (0x0010)

/* values of WFSCDMCAPS.fwRetractTransportActions */
/* values of WFSCDMCAPS.fwRetractStackerActions */
#define WFS_CDM_PRESENT (0x0001)
#define WFS_CDM_RETRACT (0x0002)
#define WFS_CDM_REJECT (0x0004)
#define WFS_CDM_NOTSUPP (0x0008)

/* values of WFSCDMCAPS.fwMoveItems */
#define WFS_CDM_FROMCU (0x0001)
#define WFS_CDM_TOCU (0x0002)
#define WFS_CDM_TOTRANSPORT (0x0004)

/* values of WFSCDMCASHUNIT.usType */
#define WFS_CDM_TYPENA (1)
#define WFS_CDM_TYPEREJECTCASSETTE (2)
#define WFS_CDM_TYPEBILLCASSETTE (3)
#define WFS_CDM_TYPECOINCYLINDER (4)
#define WFS_CDM_TYPECOINDISPENSER (5)
#define WFS_CDM_TYPERETRACTCASSETTE (6)
#define WFS_CDM_TYPECOUPON (7)
#define WFS_CDM_TYPEDOCUMENT (8)
#define WFS_CDM_TYPERECYPON (11)
#define WFS_CDM_TYPEREPCONTAINER (12)

/* values of WFSCDMCASHUNIT.usStatus */
#define WFS_CDM_STATCUOK (0)
#define WFS_CDM_STATCUFULL (1)
#define WFS_CDM_STATCUHIGH (2)
#define WFS_CDM_STATCULOW (3)
#define WFS_CDM_STATCUEMPTY (4)
#define WFS_CDM_STATCUINOP (5)
#define WFS_CDM_STATCUMISSING (6)
#define WFS_CDM_STATCUNOVAL (7)
#define WFS_CDM_STATCUNOREF (8)
#define WFS_CDM_STATCUMANIP (9)

/* values of WFSCDMMIXTYPE.usMixType */
#define WFS_CDM_MIXALGORITHM (1)
#define WFS_CDM_MIXTABLE (2)

/* values of WFSCDMMIXTYPE.usMixNumber */
#define WFS_CDM_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 WFSCDM_DISPENSE.fwPosition */
/* values of WFSCDM_CAPS.fwPositions */
/* values of WFSCDM_OUTPOS.fwPosition */
/* values of WFSCDM_TELLERPOS.fwPosition */
/* values of WFSCDM_TELLERDETAILS.fwOutputPosition */
/* values of WFSCDM_PHYSICALCU.fwPosition */

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

/* values of WFSCDM_TELLERDETAILS.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 WFSCDM_TELLERUPDATE.usAction */
#define  WFS_CDM_CREATE_TELLER  (1)
#define  WFS_CDM_MODIFY_TELLER  (2)
#define  WFS_CDM_DELETE_TELLER  (3)

/* values of WFSCDM_CUERROR.wFailure */
#define  WFS_CDM_CASHUNITEMPTY  (1)
#define  WFS_CDM_CASHUNITERROR  (2)
#define  WFS_CDM_CASHUNITFULL  (4)
#define  WFS_CDM_CASHUNITLOCKED  (5)
#define  WFS_CDM_CASHUNITINVALID  (6)
#define  WFS_CDM_CASHUNITCONFIG  (7)

/* values of lpusReason in WFS_EXEE_CDM_NOTESERROR */
#define  WFS_CDM_DOUBLENOTEDETECTED  (1)
#define  WFS_CDM_LONGNOTEDETECTED  (2)
#define  WFS_CDM_SKEWEDNOTE  (3)
#define  WFS_CDM_INCORRECTCOUNT  (4)
#define  WFS_CDM_NOTESTOOCLOSE  (5)

/* WOSA/XFS CDM Errors */
#define  WFS_ERR_CDM_INVALIDCURRENCY  (- (CDM_SERVICE_OFFSET + 0))
#define  WFS_ERR_CDM_INVALIDTELLERID  (- (CDM_SERVICE_OFFSET + 1))
#define  WFS_ERR_CDM_CASHUNITERROR  (- (CDM_SERVICE_OFFSET + 2))
#define  WFS_ERR_CDM_CASHUNITFULL  (- (CDM_SERVICE_OFFSET + 3))
#define  WFS_ERR_CDM_CASHUNITLOCKED  (- (CDM_SERVICE_OFFSET + 4))
#define  WFS_ERR_CDM_CASHUNITINVALID  (- (CDM_SERVICE_OFFSET + 5))
#define  WFS_ERR_CDM_NATIVECURRNCYMIX  (- (CDM_SERVICE_OFFSET + 6))
#define  WFS_ERR_CDM_NOTDISTINGUISHABLE  (- (CDM_SERVICE_OFFSET + 7))
#define  WFS_ERR_CDM_CDMPHYSICALCU  (- (CDM_SERVICE_OFFSET + 8))
#define  WFS_ERR_CDM_POSITIONLOCKED  (- (CDM_SERVICE_OFFSET + 9))
```c
#define WFS_ERR_CDM_SAFEDOOROPEN    (- (CDM_SERVICE_OFFSET + 10))
#define WFS_ERR_CDM_INVALIDRETRACT  (- (CDM_SERVICE_OFFSET + 11))
#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_TOOMANYCOINS    (- (CDM_SERVICE_OFFSET + 24))
#define WFS_ERR_CDM_CASHINACTIVE    (- (CDM_SERVICE_OFFSET + 25))
#define WFS_ERR_CDM_NOCASHINSTARTED(- (CDM_SERVICE_OFFSET + 26))
#define WFS_ERR_CDM_INVALIDMIXTABLE (- (CDM_SERVICE_OFFSET + 27))
#define WFS_ERR_CDM_OUTPUTPOS_NOT_EMPTY(- (CDM_SERVICE_OFFSET + 28))
#define WFS_ERR_CDM_INVALIDRETRACTPOSITION(- (CDM_SERVICE_OFFSET + 29))
#define WFS_ERR_CDM_NOTRETRACTAREA  (- (CDM_SERVICE_OFFSET + 30))
#define WFS_ERR_CDM_NOCASHBOXPRESENT(- (CDM_SERVICE_OFFSET + 33))
#define WFS_ERR_CDM_AMOUNTNOTINMIXTABLE(- (CDM_SERVICE_OFFSET + 34))
#define WFS_ERR_CDM_ITEMSNOTTAKEN   (- (CDM_SERVICE_OFFSET + 35))
#define WFS_ERR_CDM_ITEMSLEFT       (- (CDM_SERVICE_OFFSET + 36))

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

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

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

typedef struct _wfs_cdm_caps
{
    WORD        wClass;
    WORD        fwType;
    WORD        wMaxBills;
    WORD        wMaxCoins;
    WORD        wMaxDispenseItems;
    BOOL        bCompound;
    BOOL        bShutter;
    BOOL        bShutterControl;
    BOOL        bRetract;
    WORD        fwRetractAreas;
    WORD        fwRetractTransportActions;
    WORD        fwRetractStackerActions;
    BOOL        bSafeDoor;
    BOOL        bCoin;
    BOOL        bCylinders;
    BOOL        bCashBox;
    BOOL        bCashIn;
    BOOL        bRefill;
    BOOL        bAutoDeposit;
    BOOL        bVandalCheck;
}```
typedef struct _wfs_cdm_physicalcu
{
    LPSTR lpPhysicalPositionName;
    CHAR cUnitID[5];
    ULONG ulInitialCount;
    ULONG ulCount;
    ULONG ulRejectCount;
    ULONG ulMaximum;
    USHORT usPStatus;
    BOOL bHardwareSensor;
} WFSCDMPHCU, *LPWFSCDMPHCU;

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

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

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

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

typedef struct _wfs_cdm_teller_details
{
    USHORT usTellerID;
}
typedef struct _wfs_cdm_currency_exp
{
  CHAR            cCurrencyID[3];
  SHORT           sExponent;
} WFSCDMDCURRENXEP, *LPWFSCDMDCURRENXEP;

typedef struct _wfs_cdm_mix_type
{
  USHORT          usMixNumber;
  USHORT          usMixType;
  USHORT          usSubType;
  LPSTR           lpszName;
} WFSCDDMIXTYPE, *LPWFSCDDMIXTYPE;

typedef struct _wfs_cdm_mix_row
{
  ULONG           ulAmount;
  LPUSHORT        lpusMixture;
} WFSCDDMIXROW, *LPWFSCDDMIXROW;

typedef struct _wfs_cdm_mix_table
{
  USHORT          usMixNumber;
  LPSTR           lpszName;
  USHORT          usRows;
  USHORT          usCols;
  LPULONG         lpulMixHeader;
  LPWFSCDDMIXROW  *lppMixRows;
} WFSCDMIXTABLE, *LPWFSCDMIXTABLE;

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

typedef struct _wfs_cdm_present_status
{
  LPWFSCDDMIXNOMINATION  lpDenomination;
  WORD                  wPresentState;
  LPSTR                 lpszExtra;
} WFSCDDMIXPRESENTSTATUS, *LPWFSCDDMIXPRESENTSTATUS;

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

typedef struct _wfs_cdm_denominate
{
  USHORT          usTellerID;
  USHORT          usMixNumber;
  LPWFSCDDMIXNOMINATION  lpDenomination;
} WFSCDDMIXNOMINATE, *LPWFSCDDMIXNOMINATE;

typedef struct _wfs_cdm_dispense
{
  USHORT          usTellerID;
  USHORT          usMixNumber;
  USHOR
  WORD                  fwPosition;
  BOOL                  bPresent;
  LPWFSCDDMIXNOMINATION  lpDenomination;
} WFSCDDMIXDISPENSE, *LPWFSCDDMIXDISPENSE;
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;
} WFSCDMCOUNTEDPHYSUCU, *LPWFSCDMCOUNTEDPHYSUCU;

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

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

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

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

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

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

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

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

typedef struct _wfs_cdm_counts_changed {
    USHORT usCount;
    USH ORT *lpusCUNumList;
}
*/ restore alignment */
#pragma pack (pop)
#endif /* __cplusplus */
#endif /* __INC_XFSCDM__H */