CEN

WORKSHOP

AGREEMENT

CWA 16926-74

February 2020

ICS 35.200; 35.240.15; 35.240.40

English version

Extensions for Financial Services (XFS) interface specification Release 3.40 - Part 74: Cash-In Module Device Class Interface - Migration from version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

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

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

Ref. No.:CWA 16926-74:2020 E

Table of Contents

European Foreword

This CEN Workshop Agreement has been developed in accordance with the CEN-CENELEC Guide 29 “CEN/CENELEC Workshop Agreements – The way to rapid consensus” and with the relevant provisions of CEN/CENELEC Internal Regulations - Part 2. It was approved by a Workshop of representatives of interested parties on 2019-10-08, the constitution of which was supported by CEN following several public calls for participation, the first of which was made on 1998-06-24. However, this CEN Workshop Agreement does not necessarily include all relevant stakeholders.
The final text of this CEN Workshop Agreement was provided to CEN for publication on 2019-12-12.
The following organizations and individuals developed and approved this CEN Workshop Agreement:

- ATM Japan LTD
- AURIGA SPA
- BANK OF AMERICA
- CASHWAY TECHNOLOGY
- CHINAL ECTRONIC FINANCIAL EQUIPMENT SYSTEM CO.
- CIMA SPA
- CLEAR2PAY SCOTLAND LIMITED
- DIEBOLD NIXDORF
- EASTERN COMMUNICATIONS CO. LTD – EASTCOM
- FINANZ INFORMATIK
- FUJITSU FRONTECH LIMITED
- FUJITSU TECHNOLOGY
- GLORY LTD
- GRG BANKING EQUIPMENT HK CO LTD
- HESS CASH SYSTEMS GMBH & CO. KG
- HITACHI OMRON TS CORP.
- HYOSUNG TNS INC
- JIANGSU GUOGUANG ELECTRONIC INFORMATION TECHNOLOGY
- KAL
- KEBA AG
- NCR FSG
- NEC CORPORATION
- OKI ELECTRIC INDUSTRY SHENZHEN
- OKI ELECTRONIC INDUSTRY CO
- PERTO S/A

- REINER GMBH & CO KG
- SALZBURGER BANKEN SOFTWARE
- SIGMA SPA
- TEB
- ZIJIN FULCRUM TECHNOLOGY CO

It is possible that some elements of this CEN/CWA may be subject to patent rights. The CEN-CENELEC policy on patent rights is set out in CEN-CENELEC Guide 8 “Guidelines for Implementation of the Common IPR Policy on Patents (and other statutory intellectual property rights based on inventions)”. CEN shall not be held responsible for identifying any or all such patent rights.
The Workshop participants have made every effort to ensure the reliability and accuracy of the technical and non-technical content of CWA 16926-74, but this does not guarantee, either explicitly or implicitly, its correctness. Users of CWA 16926-74 should be aware that neither the Workshop participants, nor CEN can be held liable for damages or losses of any kind whatsoever which may arise from its application. Users of CWA 16926-74 do so on their own responsibility and at their own risk.

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

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

Part 2: Service Classes Definition - Programmer's Reference

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Parts 20 - 28: Reserved for future use.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Part 42: Reserved for future use.

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

Part 44: XFS MIB Application Management

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

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

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

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

Parts 49 - 60 are reserved for future use.

Part 61: Application Programming Interface (API) - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Service Provider Interface (SPI) - Programmer's Reference

Part 62: Printer and Scanning Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 63: Identification Card Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 65: PIN Keypad Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 67: Depository Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 71: Camera Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 72: Alarm Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 74: Cash-In Module Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 75: Card Dispenser Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 76: Barcode Reader Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

Part 77: Item Processing Module Device Class Interface - Migration from Version 3.30 (CWA 16926) to Version 3.40 (this CWA) - Programmer's Reference

In addition to these Programmer's Reference specifications, the reader of this CWA is also referred to a complementary document, called Release Notes. The Release Notes contain clarifications and explanations on the CWA specifications, which are not requiring functional changes. The current version of the Release Notes is available online from: https://www.cen.eu/work/Sectors/Digital_society/Pages/WSXFS.aspx.

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

1. Migration Information

XFS 3.40 has been designed to minimize backwards compatibility issues. This document highlights the changes made to the CIM device class between version 3.30 and 3.40, by highlighting the additions and deletions to the text.

2. Cash-In Module

This specification describes the functionality of an XFS compliant Cash-In Module (CIM) Service Provider. It defines the service-specific commands that can be issued to the Service Provider using the **WFSGetInfo, WFSAsyncGetInfo**, **WFSExecute** and **WFSAsyncExecute** functions.

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

This specification covers the acceptance of items. An “item” is defined as any media that can be accepted and includes coupons, documents, bills and coins. However, if coins and bills are both to be accepted separate Service Providers must be implemented for each.

All currency parameters in this specification are expressed as a quantity of minimum dispense units, as defined in the description of the WFS_INF_CIM_CURRENCY_EXP command.

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

WFS_CMD_CIM_SET_TELLER_INFO
WFS_INF_CIM_SET_TELLER_INFO

It is possible for the CIM to be part of a compound device with the Cash Dispenser Module (CDM). This CIM\CDM combination is referred to throughout this specification as a “cash recycler”. For details of the CDM interface see [Ref. 3].

If the device is a cash recycler then, if cash unit exchanges are required on both interfaces, the exchanges cannot be performed concurrently. An exchange on one interface must be complete (the WFS_CMD_CIM_END_EXCHANGE must have completed) before an exchange can start on the other interface. The WFS_ERR_CIM_EXCHANGEACTIVE error code will be returned if the correct sequence is not adhered to.

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

The event WFS_SRVE_CIM_COUNTS_CHANGED will be posted if an operation on the CDM interface affects the recycle cash unit counts which are available through the CIM interface.

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

WFS_CMD_CDM_DISPENSE
WFS_CMD_CDM_PRESENT
WFS_CMD_CDM_RETRACT
WFS_CMD_CDM_COUNT
WFS_CMD_CDM_REJECT
WFS_CMD_CDM_SET_CASH_UNIT_INFO
WFS_CMD_CDM_END_EXCHANGE
WFS_CMD_CDM_CALIBRATE_CASH_UNIT
WFS_CMD_CDM_RESET
WFS_CMD_CDM_TEST_CASH_UNITS

~~The following applies when a blacklist of items is supported via the WFS_INF_CIM_GET_BLACKLIST and WFS_CMD_CIM_SET_BLACKLIST commands. If a blacklisted item is detected the device will classify the item as a level 2 banknote and will handle the item automatically according to the local country specific note handling standard or legislation. A WFS_EXEE_CIM_INPUT_P6 and/or WFS_EXEE_CIM_INFO_AVAILABLE event will be sent if a blacklisted banknote is retained. A WFS_EXEE_CIM_INPUTREFUSE event will be sent with *lpusReason* set to WFS_CIM_INVALIDBILL if the blacklisted banknote is refused and returned to the user.~~

~~The Blacklist functionality can use a mask to specify serial numbers. The mask is defined as follows: A '?' character (0x003F) is used to represent a wildcard for a single Unicode character, and a '*' character (0x002A) is used to represent a wildcard for a single or multiple Unicode character. For example, “S8H9??16?4” would represent a match for the serial numbers “S8H9231654” and “S8H9761684”. A mask of “HD90*2” would be used in order to match serial numbers that begin with “HD90” and end with “2”, for example “HD9028882”, “HD9083276112”. Note that the blacklist mask can only use one asterisk, and if a real character is required then it must be preceded by a backslash, for example: '\\' for a backslash, '*' for an asterisk or '\?' for a question mark.~~

3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer's Reference Revision 3.~~30~~40
2. ISO 4217 at http://www.iso.org
3. XFS Cash Dispenser Device Class Interface, Programmer's Reference, Revision 3.~~30~~40
4. Paragraph 6 of the EU council regulation 1338/2001. Terms of reference for the adaptation of paragraph 6 on cash-in and cash-recycling machines (18.04.2002) at: http://www.ecb.int/pub/pdf/other/recyclingeurobanknotes2005en.pdf
5. Extensions for Financial Services (XFS) interface specification, Release 3.~~30~~40, Part 18: Item Processing Module Device Class Interface Programmer's Reference.

~~4. Legislative Note Handling Standards Support~~

~~The XFS CIM specification is designed to support legislative note handling standards that may exist in various countries and economic regions. XFS supports these note handling standards though the ability to attribute a level number to each note. The XFS classification for each level, and how each level is handled is as follows:~~

4. Note Classification

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

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

~~If a note handling standard is to be supported then this classification of levels can be used to report items which have been recognized/not recognized so that they can be processed accordingly. Where no standard is required to be supported this classification can be ignored, in which case note levels do not have to be reported.~~

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

The above classification levels can be used to support ~~standards that require~~ note handling functionality which includes:

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

A note's classification can be changed based on the note's serial number, currency and value by specifying a blacklist or classification list. A blacklist reclassifies a matching note as level 2, whereas a classification list can be used to re-classify a matching note to a lower level, including classifying a genuine note as unfit for dispensing. Once reclassified, the note will be automatically handled according to the local country specific note handling standard or legislation for the note's new note classification, including any level 2 or 3 note retention rules. Any reclassification will result in the normal events and behavior, for example a WFS_EXEE_CIM_INFO_AVAILABLE event will reflect the note's reclassification. Reclassification can be used to make dynamic changes to note handling procedures without a software upgrade, enabling functionality such as taking older notes out of circulation or handling of counterfeit notes on a local basis.

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

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

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

For example, "S8H9??16?4" would represent a match for the serial numbers "S8H9231654" and "S8H9761684". A mask of "HD90*2" would be used in order to match serial numbers that begin with "HD90" and end with "2", for example "HD9028882", "HD9083276112". Note that the mask can only use one asterisk, and if a real character is required then it must be preceded by a backslash, for example: '\\' for a backslash, '*' for an asterisk or '\?' for a question mark. Note that this flexibility means that it is possible to overlap definitions, for example "HD90*" and "HD902*" would both match on the serial number HD9028882".

5. Info Commands

5.1 WFS_INF_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 LPWFSCIMSTATUS lpStatus;

```
typedef struct _wfs_cim_status
     {
     WORD                        fwDevice;
     WORD                        fwSafeDoor;
     WORD                        fwAcceptor;
     WORD                        fwIntermediateStacker;
     WORD                        fwStackerItems;
     WORD                        fwBanknoteReader;
     BOOL                        bDropBox;
     LPWFSCIMINPOS               *lppPositions;
     LPSTR                       lpszExtra;
     DWORD                       dwGuidLights[WFS_CIM_GUIDLIGHTS_SIZE];
     WORD                        wDevicePosition;
     USHORT                      usPowerSaveRecoveryTime;
     WORD                        wMixedMode;
     WORD                        wAntiFraudModule;
     } WFSCIMSTATUS, *LPWFSCIMSTATUS;
```

fwDevice
Supplies the state of the CIM. However, an *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:

Value	Meaning
WFS_CIM_DEVONLINE	The device is online. This is returned when the acceptor is present and operational.
WFS_CIM_DEVOFFLINE	The device is offline (e.g. the operator has taken the device offline by turning a switch).
WFS_CIM_DEVPOWEROFF	The device is powered off or physically not connected.
WFS_CIM_DEVNODEVICE	The device is not intended to be there, e.g. this type of self service machine does not contain such a device or it is internally not configured.
WFS_CIM_DEVHWERROR	The device is inoperable due to a hardware error.
WFS_CIM_DEVUSERERROR	The device is present but a person is preventing proper device operation.
WFS_CIM_DEVBUSY	The device is busy and unable to process an execute command at this time.
WFS_CIM_DEVFRAUDATTEMPT	The device is present but is inoperable because it has detected a fraud attempt.
WFS_CIM_DEVPOTENTIALFRAUD	The device has detected a potential fraud attempt and is capable of remaining in service. In this case the application should make the decision as to whether to take the device offline.

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

Value	Meaning
WFS_CIM_DOORNOTSUPPORTED	Physical device has no safe door or safe door state reporting is not supported.
WFS_CIM_DOOROPEN	Safe door is open.
WFS_CIM_DOORCLOSED	Safe door is closed.
WFS_CIM_DOORUNKNOWN	Due to a hardware error or other condition, the state of the safe door cannot be determined.

fwAcceptor
Supplies the state of the acceptor cash units as one of the following values. Note that *fwAcceptor* may change value during a cash-in transaction:

Value	Meaning
WFS_CIM_ACCOK	All cash units present are in a good state.
WFS_CIM_ACCCUSTATE	One or more of the cash units is in a high, full, inoperative or manipulated condition. Items can still be accepted into at least one of the cash units.
WFS_CIM_ACCCUSTOP	Due to a cash unit failure accepting is impossible. No items can be accepted because all of the cash units are in a full, inoperative or manipulated condition. This state may also occur when a retract cash unit is full or no retract cash unit is present, or when an application lock is set on every cash unit, or when Level 2/3 notes are to be automatically retained within cash units, but all of the designated cash units for storing them are full or inoperative.
WFS_CIM_ACCCUUNKNOWN	Due to a hardware error or other condition, the state of the cash units cannot be determined.

fwIntermediateStacker
Supplies the state of the intermediate stacker as one of the following values:

Value	Meaning
WFS_CIM_ISEMPTY	The intermediate stacker is empty.
WFS_CIM_ISNOTEMPTY	The intermediate stacker is not empty.
WFS_CIM_ISFULL	The intermediate stacker is full. This may also be reported during a cash-in transaction where a limit specified by WFS_CMD_CIM_SET_CASH_IN_LIMIT has been reached.
WFS_CIM_ISUNKNOWN	Due to a hardware error or other condition, the state of the intermediate stacker cannot be determined.
WFS_CIM_ISNOTSUPPORTED	The physical device has no intermediate stacker.

fwStackerItems
This field informs the application whether items on the intermediate stacker have been in customer access. Possible values are:

Value	Meaning
WFS_CIM_CUSTOMERACCESS	Items on the intermediate stacker have been in customer access. If the device is a cash recycler then the items on the intermediate stacker may be there as a result of a previous cash-out operation.
WFS_CIM_NOCUSTOMERACCESS	Items on the intermediate stacker have not been in customer access.

WFS_CIM_ACCESSUNKNOWN	It is not known if the items on the intermediate stacker have been in customer access.
WFS_CIM_NOITEMS	There are no items on the intermediate stacker or the physical device has no intermediate stacker.

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

Value	Meaning
WFS_CIM_BNROK	The banknote reader is in a good state.
WFS_CIM_BNRINOP	The banknote reader is inoperable.
WFS_CIM_BNRUNKNOWN	Due to a hardware error or other condition, the state of the banknote reader cannot be determined.
WFS_CIM_BNRNOTSUPPORTED	The physical device has no banknote reader.

bDropBox
The drop box is an area within 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):

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

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

Value	Meaning
WFS_CIM_POSINLEFT	Left input position.
WFS_CIM_POSINRIGHT	Right input position.
WFS_CIM_POSINCENTER	Center input position.
WFS_CIM_POSINTOP	Top input position.
WFS_CIM_POSINBOTTOM	Bottom input position.
WFS_CIM_POSINFRONT	Front input position.
WFS_CIM_POSINREAR	Rear input position.
WFS_CIM_POSOUTLEFT	Left output position.
WFS_CIM_POSOUTRIGHT	Right output position.
WFS_CIM_POSOUTCENTER	Center output position.
WFS_CIM_POSOUTTOP	Top output position.
WFS_CIM_POSOUTBOTTOM	Bottom output position.
WFS_CIM_POSOUTFRONT	Front output position.
WFS_CIM_POSOUTREAR	Rear output position.

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

Value	Meaning
WFS_CIM_SHTCLOSED	The shutter is operational and is closed.
WFS_CIM_SHTOPEN	The shutter is operational and is open.
WFS_CIM_SHTJAMMED	The shutter is jammed and is not operational. The field *fwJammedShutterPosition* provides the positional state of the shutter.

WFS_CIM_SHTUNKNOWN	Due to a hardware error or other condition, the state of the shutter cannot be determined.
WFS_CIM_SHTNOTSUPPORTED	The physical device has no shutter or shutter state reporting is not supported.

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

Value	Meaning
WFS_CIM_PSEMPTY	The position is empty.
WFS_CIM_PSNOTEMPTY	The position is not empty.
WFS_CIM_PSUNKNOWN	Due to a hardware error or other condition, the state of the position cannot be determined.
WFS_CIM_PSNOTSUPPORTED	The device is not capable of reporting whether or not items are at the position.
WFS_CIM_PSFOREIGNITEMS	Foreign items have been detected in the position.

fwTransport
Specifies the state of the transport mechanism as one of the following values. The transport is defined as any area leading to or from the position:

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

fwTransportStatus
Returns information regarding items which may be on the transport. If the device is a 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:

Value	Meaning
WFS_CIM_TPSTATEMPTY	The transport is empty.
WFS_CIM_TPSTATNOTEMPTY	The transport is not empty, the items have not been in customer access.
WFS_CIM_TPSTATNOTEMPTYCUST	Items which a customer has had access to are on the transport.
WFS_CIM_TPSTATNOTEMPTY_UNK	Due to a hardware error or other condition it is not known whether there are items on the transport.
WFS_CIM_TPSTATNOTSUPPORTED	The device is not capable of reporting whether or not items are on the transport.

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

Value	Meaning
WFS_CIM_SHUTTERPOS_NOTSUPPORTED	The physical device has no shutter or the reporting of the position of a jammed shutter is not supported.
WFS_CIM_SHUTTERPOS_NOTJAMMED	The shutter is not jammed.
WFS_CIM_SHUTTERPOS_OPEN	The shutter is jammed, but fully open.
WFS_CIM_SHUTTERPOS_PARTIALLY_OPEN	The shutter is jammed, but partially open.

WFS_CIM_SHUTTERPOS_CLOSED	The shutter is jammed, but fully closed.
WFS_CIM_SHUTTERPOS_UNKNOWN	The position of the shutter is unknown.

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

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

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

Value	Meaning	Type
WFS_CIM_GUIDANCE_NOT_AVAILABLE	The status is not available.	A
WFS_CIM_GUIDANCE_OFF	The light is turned off.	A
WFS_CIM_GUIDANCE_SLOW_FLASH	The light is blinking slowly.	B
WFS_CIM_GUIDANCE_MEDIUM_FLASH	The light is blinking medium frequency.	B
WFS_CIM_GUIDANCE_QUICK_FLASH	The light is blinking quickly.	B
WFS_CIM_GUIDANCE_CONTINUOUS	The light is turned on continuous (steady).	B
WFS_CIM_GUIDANCE_RED	The light is red.	C
WFS_CIM_GUIDANCE_GREEN	The light is green.	C
WFS_CIM_GUIDANCE_YELLOW	The light is yellow.	C
WFS_CIM_GUIDANCE_BLUE	The light is blue.	C
WFS_CIM_GUIDANCE_CYAN	The light is cyan.	C
WFS_CIM_GUIDANCE_MAGENTA	The light is magenta.	C
WFS_CIM_GUIDANCE_WHITE	The light is white.	C
WFS_CIM_GUIDANCE_ENTRY	The light is in the entry state.	D
WFS_CIM_GUIDANCE_EXIT	The light is in the exit state.	D

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

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

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

wMixedMode
Reports if Mixed Media mode is active. See section WFS_CMD_CIM_SET_MODE for a description of the modes. This flag can also be set/reset by the command WFS_CMD_IPM_SET_MODE on the IPM interface. This value is one of the following values:

Value	Meaning
WFS_CIM_MIXEDMEDIANOTACTIVE	Mixed Media transactions are not supported by the device or Mixed Media mode is not activated.
WFS_CIM_IPMMIXEDMEDIA	Mixed Media mode using the CIM and IPM interfaces is activated.

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

Value	Meaning
WFS_CIM_AFMNOTSUPP	No anti-fraud module is available.
WFS_CIM_AFMOK	Anti-fraud module is in a good state and no foreign device is detected.
WFS_CIM_AFMINOP	Anti-fraud module is inoperable.
WFS_CIM_AFMDEVICEDETECTED	Anti-fraud module detected the presence of a foreign device.
WFS_CIM_AFMUNKNOWN	The state of the anti-fraud module cannot be determined.

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

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

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

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

5.2 WFS_INF_CIM_CAPABILITIES

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

Input Param None.

Output Param LPWFSCIMCAPS lpCaps;

```
typedef struct _wfs_cim_caps
     {
     WORD                        wClass;
     WORD                        fwType;
     WORD                        wMaxCashInItems;
     BOOL                        bCompound;
     BOOL                        bShutter;
     BOOL                        bShutterControl;
     BOOL                        bSafeDoor;
     BOOL                        bCashBox;
     BOOL                        bRefill;
     WORD                        fwIntermediateStacker;
     BOOL                        bItemsTakenSensor;
     BOOL                        bItemsInsertedSensor;
     WORD                        fwPositions;
     WORD                        fwExchangeType;
     WORD                        fwRetractAreas;
     WORD                        fwRetractTransportActions;
     WORD                        fwRetractStackerActions;
     LPSTR                       lpszExtra;
     DWORD                       dwGuidLights[WFS_CIM_GUIDLIGHTS_SIZE];
     DWORD                       dwItemInfoTypes;
     BOOL                        bCompareSignatures;
     BOOL                        bPowerSaveControl;
     BOOL                        bReplenish;
     WORD                        fwCashInLimit;
     WORD                        fwCountActions;
     BOOL                        bDeviceLockControl;
     WORD                        wMixedMode;
     BOOL                        bMixedDepositAndRollback;
     BOOL                        bAntiFraudModule;
     BOOL                        bDeplete;
     BOOL                        bBlacklist;
     LPDWORD                     lpdwSynchronizableCommands;
     BOOL                        bClassificationList;
     BOOL                        bPhysicalNoteList;
     } WFSCIMCAPS, *LPWFSCIMCAPS;
```

wClass
Specifies the logical service class as WFS_SERVICE_CLASS_CIM.

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

Value	Meaning
WFS_CIM_TELLERBILL	The CIM is a Teller Bill Acceptor.
WFS_CIM_SELFSERVICEBILL	The CIM is a Self-Service Bill Acceptor.
WFS_CIM_TELLERCOIN	The CIM is a Teller Coin Acceptor.
WFS_CIM_SELFSERVICECOIN	The CIM is a Self-Service Coin Acceptor.

wMaxCashInItems
Supplies the maximum number of items that can be accepted in a single WFS_CMD_CIM_CASH_IN command. This value reflects the hardware limitations of the device and therefore it does not change as part of the WFS_CMD_CIM_CASH_IN_LIMIT command.

bCompound
Specifies whether or not the logical device is part of a compound physical device.

bShutter
If this flag is TRUE then the device has a shutter and explicit shutter control through the commands WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER is supported. The definition of a shutter will depend on the h/w implementation. On some devices where items are automatically detected and accepted then a shutter is simply a latch that is opened and closed, usually under implicit control by the Service Provider. On other devices, the term shutter refers to a door, which is opened and closed to allow the customer to place the items onto a tray. If a Service Provider cannot detect when items are inserted and there is a shutter on the device, then it must provide explicit application control of the shutter.

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. In either case the WFS_CMD_CIM_PRESENT_MEDIA command may be used if the *bPresentControl* field is reported as FALSE. The *bShutterControl* field is always set to TRUE if the device has no shutter. This field applies to all shutters and all 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.

bRefill
This field is not used.

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_ITEMSTAKEN 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 actually 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. This flag should not be reported as TRUE unless item insertion can be detected.

fwPositions
Specifies the CIM input and output positions which are available as a combination of the following flags:

Value	Meaning
WFS_CIM_POSINLEFT	Left input position.
WFS_CIM_POSINRIGHT	Right input position.
WFS_CIM_POSINCENTER	Center input position.
WFS_CIM_POSINTOP	Top input position.
WFS_CIM_POSINBOTTOM	Bottom input position.
WFS_CIM_POSINFRONT	Front input position.
WFS_CIM_POSINREAR	Rear input position.
WFS_CIM_POSOUTLEFT	Left output position.
WFS_CIM_POSOUTRIGHT	Right output position.
WFS_CIM_POSOUTCENTER	Center output position.
WFS_CIM_POSOUTTOP	Top output position.
WFS_CIM_POSOUTBOTTOM	Bottom output position.
WFS_CIM_POSOUTFRONT	Front output position.
WFS_CIM_POSOUTREAR	Rear output position.

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

Value	Meaning
WFS_CIM_EXBYHAND	The CIM supports manual replenishment either by emptying the cash unit by hand or by replacing the cash unit.
WFS_CIM_EXTOCASSETTES	The CIM supports moving items from the replenishment cash unit to the bill cash units.
WFS_CIM_CLEARRECYCLER	The CIM supports the emptying of recycle cash units.
WFS_CIM_DEPOSITINTO	The CIM supports moving items from the deposit entrance to the bill cash units.

fwRetractAreas
Specifies the areas to which items may be retracted. If the device does not have a retract capability this field will be WFS_CIM_RA_NOTSUPP. Otherwise this field will be set to a combination of the following flags:

Value	Meaning
WFS_CIM_RA_RETRACT	Items may be retracted to a retract cash unit.
WFS_CIM_RA_REJECT	Items may be retracted to a reject cash unit.
WFS_CIM_RA_TRANSPORT	Items may be retracted to the transport.
WFS_CIM_RA_STACKER	Items may be retracted to the intermediate stacker.
WFS_CIM_RA_BILLCASSETTES	Items may be retracted to item cassettes, i.e. cash-in and recycle cash units.
WFS_CIM_RA_CASHIN	Items may be retracted to a cash-in cash unit.

fwRetractTransportActions
Specifies the actions which may be performed on items which have been retracted to the transport. If the device does not have the capability to retract items to or from the transport this field will be WFS_CIM_NOTSUPP. Otherwise this field will be set to a combination of the following flags:

Value	Meaning
WFS_CIM_PRESENT	The items may be moved to the exit position.
WFS_CIM_RETRACT	The items may be retracted to a retract cash unit.
WFS_CIM_REJECT	The items may be retracted to a reject cash unit.
WFS_CIM_BILLCASSETTES	The items may be retracted to item cassettes, i.e. cash-in and recycle cash units.
WFS_CIM_CASHIN	The items may be retracted to a cash-in cash unit.

fwRetractStackerActions
Specifies the actions which may be performed on items which have been retracted to the stacker. If the device does not have the capability to retract items to or from the stacker this field will be WFS_CIM_NOTSUPP. Otherwise this field will be set to a combination of the following flags:

Value	Meaning
WFS_CIM_PRESENT	The items may be moved to the exit position.
WFS_CIM_RETRACT	The items may be retracted to a retract cash unit.
WFS_CIM_REJECT	The items may be retracted to a reject cash unit.
WFS_CIM_BILLCASSETTES	The items may be retracted to item cassettes, i.e. cash-in and recycle cash units.
WFS_CIM_CASHIN	The items may be retracted to a cash-in cash unit.

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

The parameter that reports ~~if a legislative note handling standard is supported~~how notes are classified and handled is reported in *lpszExtra* as follows. If level 2/3 notes are not to be returned to the customer by these rules, they will not be returned regardless of whether their specific note type is configured to not be accepted by WFS_CMD_CIM_CONFIGURE_NOTETYPES:

P6=1	~~A note handling standard is supported~~Notes are classified as level 1, 2, 3 or 4 and only level 2 notes will not be returned to the customer in a cash-in transaction.
P6=2	Notes are classified as level 1, 2, 3 or 4 and level 2 and level 3 notes will not be returned to the customer in a cash-in transaction.

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

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

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

Value	Meaning	Type
WFS_CIM_GUIDANCE_NOT_AVAILABLE	There is no guidance light control available at this position.	A
WFS_CIM_GUIDANCE_OFF	The light can be off.	B
WFS_CIM_GUIDANCE_SLOW_FLASH	The light can blink slowly.	B
WFS_CIM_GUIDANCE_MEDIUM_FLASH	The light can blink medium frequency.	B
WFS_CIM_GUIDANCE_QUICK_FLASH	The light can blink quickly.	B
WFS_CIM_GUIDANCE_CONTINUOUS	The light can be continuous (steady).	B
WFS_CIM_GUIDANCE_RED	The light can be red.	C
WFS_CIM_GUIDANCE_GREEN	The light can be green.	C
WFS_CIM_GUIDANCE_YELLOW	The light can be yellow.	C
WFS_CIM_GUIDANCE_BLUE	The light can be blue.	C
WFS_CIM_GUIDANCE_CYAN	The light can be cyan.	C
WFS_CIM_GUIDANCE_MAGENTA	The light can be magenta.	C
WFS_CIM_GUIDANCE_WHITE	The light can be white.	C
WFS_CIM_GUIDANCE_ENTRY	The light can be in the entry state.	D
WFS_CIM_GUIDANCE_EXIT	The light can be in the exit state.	D

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

Value	Meaning
WFS_CIM_GUIDANCE_POSINNULL	The default input position.
WFS_CIM_GUIDANCE_POSINLEFT	Left input position.
WFS_CIM_GUIDANCE_POSINRIGHT	Right input position.

WFS_CIM_GUIDANCE_POSINCENTER	Center input position.
WFS_CIM_GUIDANCE_POSINTOP	Top input position.
WFS_CIM_GUIDANCE_POSINBOTTOM	Bottom input position.
WFS_CIM_GUIDANCE_POSINFRONT	Front input position.
WFS_CIM_GUIDANCE_POSINREAR	Rear input position.
WFS_CIM_GUIDANCE_POSOUTLEFT	Left output position.
WFS_CIM_GUIDANCE_POSOUTRIGHT	Right output position.
WFS_CIM_GUIDANCE_POSOUTCENTER	Center output position.
WFS_CIM_GUIDANCE_POSOUTTOP	Top output position.
WFS_CIM_GUIDANCE_POSOUTBOTTOM	Bottom output position.
WFS_CIM_GUIDANCE_POSOUTFRONT	Front output position.
WFS_CIM_GUIDANCE_POSOUTREAR	Rear output position.
WFS_CIM_GUIDANCE_POSOUTNULL	The default output position.

dwItemInfoTypes
Specifies the types of information that can be retrieved through the WFS_INF_CIM_GET_ITEM_INFO command ~~as~~. This field will either be set to WFS_CIM_ITEM_NOTSUPP or a combination of the following flags:

Value	Meaning
WFS_CIM_ITEM_SERIALNUMBER	Serial Number of the item.
WFS_CIM_ITEM_SIGNATURE	Signature of the item.
WFS_CIM_ITEM_IMAGEFILE	Image file of the item.

bCompareSignatures
Specifies if the Service Provider has the ability to compare signatures through command WFS_CMD_CIM_COMPARE_P6_SIGNATURE. If this field is set to FALSE, the WFS_CMD_CIM_COMPARE_P6_SIGNATURE command returns WFS_ERR_UNSUPP_COMMAND.

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

bReplenish
If set to TRUE the WFS_INF_CIM_REPLENISH_TARGET and WFS_CMD_CIM_REPLENISH commands are supported. If set to FALSE the WFS_INF_CIM_REPLENISH_TARGET command returns WFS_ERR_UNSUPP_CATEGORY and the WFS_CMD_CIM_REPLENISH command returns WFS_ERR_UNSUPP_COMMAND.

fwCashInLimit
Specifies whether the cash-in limitation is supported or not for the WFS_CMD_CIM_SET_CASH_IN_LIMIT command. If the device does not have the capability to limit the amount or the number of items during cash-in operations then this field will be WFS_CIM_LIMITNOTSUPP. Otherwise this field will be set to a combination of the following flags:

Value	Meaning
WFS_CIM_LIMITBYTOTALITEMS	The number of successfully processed cash-in items can be limited by specifying the total number of items.
WFS_CIM_LIMITBYAMOUNT	The number of successfully processed cash-in items can be limited by specifying the ~~total~~maximum amount of a specific currency.
WFS_CIM_LIMITMULTIPLE	WFS_CMD_CIM_SET_CASH_IN_LIMIT may be called multiple times in a cash-in transaction to update previously specified amount limits. Only valid if combined with WFS_CIM_LIMITBYAMOUNT.

WFS_CIM_LIMITREFUSEOTHER	If multiple currencies can be accepted and an amount limit is specified for one or more currencies, any other unspecified currencies are refused. If not specified, there is no amount limit for unspecified currencies. Only valid if specified with WFS_CIM_LIMITBYAMOUNT.

fwCountActions
Specifies the count action supported by the WFS_CMD_CIM_CASH_UNIT_COUNT command. If the device does not support counting then this field will be WFS_CIM_COUNTNOTSUPP. Otherwise this field will be set to a combination of the following flags:

Value	Meaning
WFS_CIM_COUNTINDIVIDUAL	The counting of individual cash units via the input structure of the WFS_CMD_CIM_CASH_UNIT_COUNT command is supported.
WFS_CIM_COUNTALL	The counting of all cash units via the NULL pointer input parameter of the WFS_CMD_CIM_CASH_UNIT_COUNT command is supported.

bDeviceLockControl
Specifies whether the CIM supports physical lock/unlock control of the CIM device and/or the cash units. If this value is set to TRUE, the device and/or the cash units can be locked and unlocked by the WFS_CMD_CIM_DEVICE_LOCK_CONTROL command, and the lock status can be retrieved by the WFS_INF_CIM_DEVICELOCK_STATUS command. If this value is set to FALSE, the CIM will not support the physical lock/unlock control of the CIM device or the cash units; the WFS_CMD_CIM_DEVICE_LOCK_CONTROL command will return WFS_ERR_UNSUPP_COMMAND and the WFS_INF_CIM_DEVICELOCK_STATUS command will return WFS_ERR_UNSUPP_CATEGORY.

wMixedMode
Specifies whether the device supports accepting and processing items other than the types defined in the CIM specification. For a description of Mixed Media transactions see section ATM Mixed Media Transaction Flow – Application Guidelines. If the device does not support Mixed Media processing this field will be WFS_CIM_MIXEDMEDIANOTSUPP. Otherwise this field will be set to the following value:

Value	Meaning
WFS_CIM_IPMMIXEDMEDIA	Mixed Media transactions are supported using the CIM and IPM interfaces.

bMixedDepositAndRollback
Specifies whether the device can deposit one type of media and rollback the other in the same Mixed Media transaction. Where *bMixedDepositAndRollback* is TRUE the Service Provider can accept WFS_CMD_CIM_CASH_IN_END and WFS_CMD_IPM_MEDIA_IN_ROLLBACK or WFS_CMD_CIM_CASH_IN_ROLLBACK and WFS_CMD_IPM_MEDIA_IN_END to complete the current transaction. This value can only be TRUE where *wMixedMode* == WFS_CIM_IPMMIXEDMEDIA. When *bMixedDepositAndRollback* is FALSE applications must either deposit or return ALL items to complete a transaction. Where Mixed Media transactions are not supported *bMixedDepositAndRollback* is FALSE.

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

bDeplete
If set to TRUE the WFS_CMD_CIM_DEPLETE command is supported. If set to FALSE the WFS_CMD_CIM_DEPLETE command returns WFS_ERR_UNSUPP_COMMAND.

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

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

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

bPhysicalNoteList
Specifies whether the Service Provider supports note number lists on physical cash units (see *lpszExtra* in WFSCIMPHCU) This can either be TRUE if the Service Provider has the capability or FALSE if it does not.

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

Comments Applications which rely on the *lpszExtra* field may not be device or vendor-independent. The table below defines the valid combinations of *bShutter, bShutterControl* and WFSCIMPOSCAPS.*bPresentControl.*

bShutter	*bShutterControl*	WFSCIMPOSCAPS .*bPresentControl*	Description
TRUE	TRUE	TRUE	Service Provider implicitly opens the shutter, presents items and closes the shutter when all items are taken.
TRUE	TRUE	FALSE	Service Provider implicitly opens the shutter for input. Application required to present items using WFS_CMD_CIM_PRESENT_MEDIA.
TRUE	FALSE	TRUE	Application is required to present items using WFS_CMD_CIM_OPEN_SHUTTER and then call WFS_CMD_CIM_CLOSE_SHUTTER when all items are taken.
TRUE	FALSE	FALSE	Application is required to present items either by using WFS_CMD_CIM_PRESENT_MEDIA~~,~~ or ~~alternatively,~~ by using~~,~~ WFS_CMD_CIM_OPEN_SHUTTER and then WFS_CMD_CIM_CLOSE_SHUTTER when all items are taken.
FALSE	TRUE	TRUE	Service Provider implicitly opens the shutter, presents items and closes the shutter when all items taken.
FALSE	TRUE	FALSE	Service Provider implicitly opens the shutter for input. Application required to present items using WFS_CMD_CIM_PRESENT_MEDIA.
FALSE	FALSE	TRUE	Not Supported.
FALSE	FALSE	FALSE	Application required to present items using WFS_CMD_CIM_PRESENT_MEDIA.

5.3 WFS_INF_CIM_CASH_UNIT_INFO

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

Where a logical cash unit or recycle unit is configured but there is no corresponding physical cash unit currently present in the device, information about the missing cash unit or recycle unit will still be returned in the *lppCashIn* field of the output parameter. The status of the cash unit or recycle unit will be reported as WFS_CIM_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 *lpCashInfo* will reflect the number of logical cash units in the CIM. That is, if a system contains four physical cash units but two of these are treated as one logical cash unit, *lpCashInfo* 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 WFSCIMCASHUNIT 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 *lpCashInfo* will again reflect the number of logical cash units in the CIM.

Counts

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

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

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

Exchanges

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

On recycle and retract cash units the counts and status reflect the physical status of the cassette and therefore are consistently reported on both the CDM and CIM interfaces. When a value is changed through an exchange on one interface it is also changed on the other.

Recyclers

The CIM interface reports all cash units including cash-out only cash units. The CDM interface does not report cash-in only cash units but does report cash units used on both interfaces, i.e. recycle cash units (WFS_CIM_TYPERECYCLING) and reject/retract cash units (WFS_CIM_TYPEREJECT / WFS_CIM_TYPERETRACTCASSETTE).

Input Param None.

Output Param LPWFSCIMCASHINFO lpCashInfo;

```
typedef struct _wfs_cim_cash_info
     {
     USHORT                      usCount;
     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;
     DWORD                      fwType;
     DWORD                      fwItemType;
     CHAR                       cUnitID[5];
     CHAR                       cCurrencyID[3];
     ULONG                      ulValues;
     ULONG                      ulCashInCount;
     ULONG                      ulCount;
     ULONG                      ulMaximum;
     USHORT                     usStatus;
     BOOL                       bAppLock;
     LPWFSCIMNOTENUMBERLIST     lpNoteNumberList;
     USHORT                     usNumPhysicalCUs;
     LPWFSCIMPHCU               *lppPhysical;
     LPSTR                      lpszExtra;
     LPUSHORT                   lpusNoteIDs;
     WORD                       usCDMType;
     LPSTR                      lpszCashUnitName;
     ULONG                      ulInitialCount;
     ULONG                      ulDispensedCount;
     ULONG                      ulPresentedCount;
     ULONG                      ulRetractedCount;
     ULONG                      ulRejectCount;
     ULONG                      ulMinimum;
     } 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 as one of the following values:

Value	Meaning
WFS_CIM_TYPERECYCLING	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.
WFS_CIM_TYPECASHIN	Cash-in cash unit.
WFS_CIM_TYPEREPCONTAINER	Replenishment container. A cash unit can be refilled from or emptied to a replenishment container.
WFS_CIM_TYPERETRACTCASSETTE	Retract cash unit.
WFS_CIM_TYPEREJECT	Reject cash unit.
WFS_CIM_TYPECDMSPECIFIC	A cash unit that is only applicable to the CDM interface. This value is used to report CDM cash units of the following types: WFS_CDM_TYPENA, WFS_CDM_TYPEBILLCASSETTE, WFS_CDM_TYPECOINCYLINDER, WFS_CDM_TYPECOINDISPENSER, WFS_CDM_TYPECOUPON and WFS_CDM_TYPEDOCUMENT. See the *usCDMType* field for details of the cash unit type.

fwItemType
Specifies the type of items the cash unit takes as a combination of the following flags. The table in the Comments section of this command defines how to interpret the combination of these flags:

Value	Meaning
WFS_CIM_CITYPALL	The cash~~ in~~ unit takes all fit banknote types. ~~If a note handling standard is supported, then these~~These are level 4 notes which are fit for recycling.
WFS_CIM_CITYPUNFIT	The cash~~ in~~ unit takes all unfit banknotes. ~~If a note handling standard is supported, then these~~These are level 4 notes which are unfit for recycling.
WFS_CIM_CITYPINDIVIDUAL	The cash~~ in unit or recycle cash~~ unit takes all types of fit banknotes specified in an individual list. ~~If a note handling standard is supported, then these~~These are level 4 notes which are fit for recycling.
WFS_CIM_CITYPLEVEL1	Level 1 note types are stored in this cash unit.
WFS_CIM_CITYPLEVEL2	If ~~a note handling standard is supported~~notes can be classified as level 2, then level 2 note types are stored in this cash~~ in~~ unit.
WFS_CIM_CITYPLEVEL3	If ~~a note handling standard is supported~~notes can be classified as level 3, then level 3 note types are stored in this cash~~ in~~ unit.
WFS_CIM_CITYPIPM	The cash~~ in~~ unit can accept items on the IPM interface.
WFS_CIM_CITYPUNFITINDIVIDUAL	The cash unit takes all types of unfit banknotes specified in an individual list. These are level 4 notes which are unfit for recycling.

Support for classifying validated notes as 'unfit' is hardware dependent. On h/w that cannot classify notes as 'unfit', all validated banknotes will be treated as 'fit' and accepted by cash units of type WFS_CIM_CITYPALL and/or WFS_CIM_CITYPINDIVIDUAL. On such h/w the value WFS_CIM_CITYPUNFIT will not be used.

On h/w that can classify notes as 'unfit', validated 'fit' banknotes will be accepted by cash units of type WFS_CIM_CITYPALL and/or WFS_CIM_CITYPINDIVIDUAL. If the cash unit is configured as a combination of WFS_CIM_CITYPALL or WFS_CIM_CITYPINDIVIDUAL with WFS_CIM_CITYPUNFIT then the cash unit accepts valid 'fit' and 'unfit' banknote types. If the cash unit is configured as a combination of WFS_CIM_CITYPINDIVIDUAL with WFS_CIM_CITYPUNFITINDIVIDUAL then the cash unit accepts valid 'fit' and 'unfit' banknote types of the note types specified in an individual list.

This value is zero for cash units that cannot accept media items, i.e. cash units that can only dispense, or for cash units that are configured not to accept any items. It may be possible to use the command WFS_CMD_CIM_CONFIGURE_CASH_IN_UNITS to configure the cash unit to accept media.

cUnitID
The Cash Unit Identifier.

cCurrencyID
A three character array storing the ISO format currency ID [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 *usStatus* field for this cash unit is WFS_CIM_STATCUNOVAL it is the responsibility of the application to assign a value to this field. This value is persistent.

ulValues
Supplies the value of a single item in the cash unit. This value is expressed in minimum dispense units (see section WFS_INF_CIM_CURRENCY_EXP). If the *cCurrencyID* field for this cash unit is an array of three ASCII 0x20h characters or the cash unit is configured to accept more than one denomination of note then this field will contain zero. The value of the notes stored in the cash unit can be calculated from the contents of *lpNoteNumberList* and the data returned from the WFS_INF_CIM_BANKNOTE_TYPES command. If the *usStatus* field for this cash unit is WFS_CIM_STATCUNOVAL it is the responsibility of the application to assign a value to this field. This value is persistent.

ulCashInCount
Count of items that have entered the logical cash unit. This counter is incremented whenever an item enters a physical cash unit that belongs to this logical cash unit for any reason~~.~~, unless it originated from this cash unit but was returned without being accessible to a customer. For a retract cash unit this value represents the total number of items of all types in the cash unit, or if the device cannot count items during a retract operation this value will be zero. If *fwType* is WFS_CIM_TYPECDMSPECIFIC then this value is zero. This value is persistent.

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

For all cash units except retract cash units (*fwType* is not WFS_CIM_TYPERETRACTCASSETTE) this value reports the total number of banknotes, checks or coins of all types in the cash unit.

For cash units supporting the *fwItemType* WFS_CIM_CITYPIPM the number of banknotes or coins contained in the cash unit can be determined from *lpNoteNumberList.*

If the cash unit is a recycle cash unit (*fwType* is WFS_CIM_TYPERECYCLING) then this value may not be the same as the value of *ulCashInCount.* This value will be decremented as a result of a dispense transaction on the CDM interface. During dispense transactions on the CDM, this value includes any items that have been dispensed but not yet presented to the customer. This count is only decremented when these items are either known to be in customer access, successfully rejected or moved to another cash unit.

If the cash unit is a retract cash unit (*fwType* is WFS_CIM_TYPERETRACTCASSETTE) then this value will not normally be the same as the value of *ulCashInCount.* This value specifies the number of retract operations (CIM commands, CDM commands and error recovery) which result in items entering the cash unit.

If the cash unit is CDM specific (*fwType* is WFS_CIM_TYPECDMSPECIFIC) then this value will be reported as defined in the CDM interface specification.

ulMaximum
When the *ulCount* reaches this value the threshold event WFS_USRE_CIM_CASHUNITTHRESHOLD (WFS_CIM_STATCUHIGH) will be generated. If this value is non-zero then hardware sensors in the device do not trigger threshold events. If this value is zero then hardware sensors will trigger threshold events if *bHardwareSensors* is TRUE.

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

Value	Meaning
WFS_CIM_STATCUOK	The cash unit is in a good state.
WFS_CIM_STATCUFULL	The cash unit is full. This value is not used for CDM specific cash units (*fwType* == WFS_CIM_TYPECDMSPECIFIC).
WFS_CIM_STATCUHIGH	The cash unit is almost full (i.e. reached or exceeded the threshold defined by *ulMaximum*). This value is not used for CDM specific cash units (*fwType* == WFS_CIM_TYPECDMSPECIFIC).

WFS_CIM_STATCULOW	The cash unit is almost empty (i.e. reached or below the threshold defined by *ulMinimum*). This value is only reported for ~~CDM specific~~cash units which can dispense media items. It is not mandatory to report this for recycle cash units (*fwType* == WFS_CIM_~~TYPECDMSPECIFIC~~TYPERECYCLING).
WFS_CIM_STATCUEMPTY	The cash unit is empty. On a dispensing cash unit on a recycler this can be caused by insufficient items in the cash unit preventing further dispense operations.
WFS_CIM_STATCUINOP	The cash unit is inoperative.
WFS_CIM_STATCUMISSING	The cash unit is missing.
WFS_CIM_STATCUNOVAL	The values of the specified cash unit are not available. This can be the case when the cash unit is changed without using the operator functions.
WFS_CIM_STATCUNOREF	There is no reference value available for the notes in this cash unit. The cash unit has not been configured. This value has no meaning on the CIM and is not used.
WFS_CIM_STATCUMANIP	The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. Items cannot be accepted into this cash unit.

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. This value is persistent.

lpNoteNumberList
Pointer to a WFSCIMNOTENUMBERLIST structure. The content of this structure is persistent.

If the cash unit is a CDM specific cash unit (*fwType* == WFS_CIM_TYPECDMSPECIFIC) with *usCDMType* == WFS_CDM_TYPEBILLCASSETTE this pointer will be NULL.

If the cash unit is **not** a retract cash unit (*fwType* is not WFS_CIM_TYPERETRACTCASSETTE), then the *lpNoteNumberList* will point to the list of cash items inside the cash unit. Additionally if the contents of the cash unit are not known then this pointer will be NULL.

If the cash unit is a retract cash unit (*fwType* == WFS_CIM_TYPERETRACTCASSETTE) this pointer will be NULL except for the following cases:

- ~~If a note handling standard is supported and~~If the retract cash unit is configured to accept level 2 notes then the number and type of level 2 notes is returned in the *lpNoteNumberList* and *ulCount* contains the number of retract operations. *ulCashInCount* contains the actual number of level 2 notes.

- If items are recognized during retract operations then the number and type of notes retracted is returned in *lpNoteNumberList* and *ulCount* contains the number of retract operations. *ulCashInCount* contains the actual number of retracted items.

If both cases apply then the number and type of level 2 notes and notes retracted is returned in the *lpNoteNumberList* and *ulCount* contains the number of retract operations. *ulCashInCount* contains the actual number of level 2 notes and retracted items.

```
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:

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

usNoteID
Identification of note type. The Note ID represents the note identifiers reported by the WFS_INF_CIM_BANKNOTE_TYPES command. If this value is zero then the note type is unknown.

ulCount
Actual count of cash items. The value is incremented each time cash items are moved to a cash unit by a **WFSExecute** command. In the case of recycle cash units this count is decremented as defined in the description of the logical *ulCount* field.

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 WFSCIMPHCU structures:

```
typedef struct _wfs_cim_physicalcu
     {
     LPSTR                      lpPhysicalPositionName;
     CHAR                       cUnitID[5];
     ULONG                      ulCashInCount;
     ULONG                      ulCount;
     ULONG                      ulMaximum;
     USHORT                     usPStatus;
     BOOL                       bHardwareSensors;
     LPSTR                      lpszExtra;
     ULONG                      ulInitialCount;
     ULONG                      ulDispensedCount;
     ULONG                      ulPresentedCount;
     ULONG                      ulRetractedCount;
     ULONG                      ulRejectCount;
     } WFSCIMPHCU, *LPWFSCIMPHCU;
```

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 or IPM to identify shared cash units/media bins.

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

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

ulCount
As defined by the logical *ulCount* description but applies to a single physical cash unit. The one exception is that during dispense transactions on the CDM, this value does not include any items that have been dispensed but not yet presented. This value is persistent.

ulMaximum
Maximum count of items in the physical cash unit. No threshold event will be generated when this value is reached. This value is persistent. This field is deprecated. The value for *ulMaximum* is reported using the WFS_INF_CIM_CASH_UNIT_CAPABILITIES command.

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

Value	Meaning
WFS_CIM_STATCUOK	The cash unit is in a good state.
WFS_CIM_STATCUFULL	The cash unit is full. This value is not used for CDM specific cash units (*fwType* == WFS_CIM_TYPECDMSPECIFIC).
WFS_CIM_STATCUHIGH	The cash unit is almost full (reached or exceeded the threshold defined by *ulMaximum* in physical structure). This value is not used for CDM specific cash units (*fwType* == WFS_CIM_TYPECDMSPECIFIC).
WFS_CIM_STATCULOW	The cash unit is almost empty. This value is only reported for ~~CDM specific~~cash units which can dispense media items. It is not mandatory to report this for recycle cash units (*fwType* == WFS_CIM_~~TYPECDMSPECIFIC~~TYPERECYCLING).
WFS_CIM_STATCUEMPTY	The cash unit is empty. On a dispensing cash unit on a recycler this can be caused by insufficient items in the cash unit preventing further dispense operations.
WFS_CIM_STATCUINOP	The cash unit is inoperative.
WFS_CIM_STATCUMISSING	The cash unit is missing (the cash unit has been removed and is physically not present in the machine).
WFS_CIM_STATCUNOVAL	The values of the specified cash unit are not available.
WFS_CIM_STATCUNOREF	There is no reference value available for the notes in this cash unit. The cash unit has not been configured. This value is only reported for CDM specific cash units (*fwType* == WFS_CIM_TYPECDMSPECIFIC).
WFS_CIM_STATCUMANIP	The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state.

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. This field is deprecated. The value for *bHardwareSensors* is reported using the WFS_INF_CIM_CASH_UNIT_CAPABILITIES command.

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

If the *bPhysicalNoteList* capability is TRUE, the breakdown of notes within the physical cash unit may be specified or reported using an optional string of the following format which can be mapped onto a WFSCIMNOTENUMBERLIST structure. It is not mandatory to specify this string during a replenishment operation even if the *bPhysicalNoteList* capability is TRUE. See Rules for Cash Unit Exchange for an example and details of how this can be used:

NOTENUMBERLIST=<*semi-colon separated list of note numbers*>

Where each note number (compare with WFSCIMNOTENUMBER) is represented by

<Note ID>,<Count>

Where

<Note ID> is the Note ID in decimal (see *WFSCIMNOTENUMBER::usNoteID*)

<Count> is the number of notes in decimal of Note ID *<Note ID>* (see *WFSCIMNOTENUMBER::ulCount*)

For example if a physical cash unit contains 30 notes of note ID 1 and 100 notes of note ID 5, this would be represented with the following key/value pair

NOTENUMBERLIST=1,30;5,100

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

ulDispensedCount
The number of items dispensed from this physical cash unit. This value is persistent. See the CDM interface specification for details.

ulPresentedCount
The number of items from this physical cash unit that have been presented to the customer by the CDM interface. This value is persistent. See the CDM interface specification for details.

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

ulRejectCount
The number of items from this physical cash unit which ~~are in a reject bin.~~have been rejected. This value is persistent. See the CDM interface specification for details.

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

lpusNoteIDs
Pointer to a zero-terminated list of unsigned shorts which contains the note IDs of the banknotes the cash-in cash unit or recycle cash unit can take. This field only applies to WFS_CIM_CITYPINDIVIDUAL cassette types. If there are no note IDs defined for the cassette or the cassette is not defined as WFS_CIM_CITYPINDIVIDUAL then *lpusNoteIDs* will contain NULL.

usCDMType
The type of cash unit reported for the corresponding cash unit on the CDM interface. See the CDM interface specification for details. For CIM only cash units this value is zero.

lpszCashUnitName
An application defined name to help identify the content of the cash unit. This value can be NULL.

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

ulDispensedCount
The number of items dispensed from all the physical cash units associated with this logical cash unit. This value is persistent. See the CDM interface specification for details.

ulPresentedCount
The number of items from all the physical cash units associated with this logical cash unit that have been presented to the customer by the CDM interface. This value is persistent. See the CDM interface specification for details.

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

ulRejectCount
The number of items from this logical cash unit which ~~are in a reject bin.~~have been rejected. This value is persistent. See the CDM interface specification for details.

ulMinimum
This field is only applicable to CDM cash units which can dispense media items. This value is persistent. See the CDM interface specification for details.

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

Comments The following table defines the interpretation of the *fwItemType* flag for single values and a sub-set of possible combinations (many of which may not actually be possible on physical hardware implementations). The check mark means that the corresponding flag is set, empty means that the corresponding flag is not set.

For a definition of the terms 'fit' and 'unfit' see the description of *fwItemType* itself. The combinations not included in this table can be interpolated from this table.

ALL	UNFIT	INDIVIDUAL	LEVEL 3	LEVEL 2	LEVEL 1	UNFITINDIVIDUAL	Description
√							Fit notes for all note ids
	√						Unfit notes for all note ids
		√					Fit notes from the Individual note list
			√				Level 3 notes for all note ids
				√			Level 2 notes for all note ids
√	√						Fit notes for all note ids & unfit notes for all note ids
√			√				Fit notes for all note ids & level 3 notes for all note ids
√				√			Fit notes for all note ids & level 2 notes for all note ids
√			√	√			Fit notes for all note ids & level 3 notes for all note ids & level 2 notes for all note ids
√	√		√	√			Fit notes for all note ids & unfit notes for all note ids & level 3 notes for all note ids & level 2 notes for all note ids
	√	√					Fit notes from the Individual note list & unfit notes for all note ids
		√	√				Fit notes from the Individual note list & level 3 notes for all note ids.
		√		√			Fit notes from the Individual note list &

							level 2 notes for all note ids.
		√	√	√			Fit notes from the Individual note list & level 3 notes for all note ids & level 2 notes for all note ids.
	√	√	√	√			Fit notes from the Individual note list & unfit notes for all note ids & level 3 notes for all note ids & level 2 notes for all note ids.
					√		Unrecognized notes.
		√				√	Fit & unfit notes from the individual note list
						√	Unfit notes from the individual note list

Note: WFS_CIM_CITYPALL always overrides WFS_CIM_CITYPINDIVIDUAL when these values are combined.
WFS_CIM_CITYPIPM can be combined with any other combination and indicates non-note items can be stored in this cash unit.

WFS_CIM_CITYPUNFIT always overrides WFS_CIM_CITYPUNFITINDIVIDUAL when these values are combined.

5.4 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 0x20 characters. In this case information on all currencies will be returned.

Output Param LPWFSCIMTELLERDETAILS *lppTellerDetails;

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

```
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:

Value	Meaning
WFS_CIM_POSNULL	No position is assigned to the teller.
WFS_CIM_POSINLEFT	The left position is assigned to the teller.
WFS_CIM_POSINRIGHT	The right position is assigned to the teller.
WFS_CIM_POSINCENTER	The center position is assigned to the teller.
WFS_CIM_POSINTOP	The top position is assigned to the teller.
WFS_CIM_POSINBOTTOM	The bottom position is assigned to the teller.
WFS_CIM_POSINFRONT	The front position is assigned to the teller.
WFS_CIM_POSINREAR	The rear position is assigned to the teller.

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

Value	Meaning
WFS_CIM_POSNULL	No position is assigned to the teller.
WFS_CIM_POSOUTLEFT	The left position is assigned to the teller.
WFS_CIM_POSOUTRIGHT	The right position is assigned to the teller.
WFS_CIM_POSOUTCENTER	The center position is assigned to the teller.
WFS_CIM_POSOUTTOP	The top position is assigned to the teller.
WFS_CIM_POSOUTBOTTOM	The bottom position is assigned to the teller.
WFS_CIM_POSOUTFRONT	The front position is assigned to the teller.
WFS_CIM_POSOUTREAR	The rear position is assigned to the teller.

lppTellerTotals
Pointer to a NULL-terminated array of pointers to WFSCIMTELLERTOTALS structures.

```
typedef struct _wfs_cim_teller_totals
     {
     CHAR                      cCurrencyID[3];
     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 section WFS_INF_CIM_CURRENCY_EXP).

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

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

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

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

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

Value	Meaning
WFS_ERR_CIM_INVALIDCURRENCY	Specified currency not currently available.
WFS_ERR_CIM_INVALIDTELLERID	Invalid teller ID.

Comments None.

5.5 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 LPWFSCIMCURRENCYEXP *lppCurrencyExp;

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

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

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

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

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

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

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

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

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

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

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

5.6 WFS_INF_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 LPWFSCIMNOTETYPELIST lpNoteTypeList;

```
typedef struct _wfs_cim_note_type_list
     {
     USHORT                      usNumOfNoteTypes;
     LPWFSCIMNOTETYPE            *lppNoteTypes;
     } WFSCIMNOTETYPELIST, *LPWFSCIMNOTETYPELIST;
```

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 WFSCIMNOTETYPE structures:

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

usNoteID
Identification of note type.

cCurrencyID
Currency ID in ISO 4217 format [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 is, the newer the release. Zero means that there is only one release of that banknote type. This value has not been standardized and therefore a release number of the same banknote will not necessarily have the same value in different systems.

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 unless it must be retained by note classification rules.

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

Comments None.

5.7 WFS_INF_CIM_CASH_IN_STATUS

Description This command is used to get information about the status of the currently active cash-in transaction or in the case where no cash-in transaction is active the status of the most recently ended cash-in transaction. This value is persistent and is valid until the next command WFS_CMD_CIM_CASH_IN_START.

Input Param None.

Output Param LPWFSCIMCASHINSTATUS lpCashInStatus;

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

wStatus
Status of the currently active or most recently ended cash-in transaction. Possible values are:

Value	Meaning
WFS_CIM_CIOK	The cash-in transaction is complete and has ended with a WFS_CMD_CIM_CASH_IN_END command call.
WFS_CIM_CIROLLBACK	The cash-in transaction was has ended with a WFS_CMD_CIM_CASH_IN_ROLLBACK command call.
WFS_CIM_CIACTIVE	There is a cash-in transaction active. See the WFS_CMD_CIM_CASH_IN_START command description for a definition of an active cash-in transaction.
WFS_CIM_CIRETRACT	The cash-in transaction ended with a WFS_CMD_CIM_RETRACT command call, or a retract command call on a compound device class.
WFS_CIM_CIUNKNOWN	The state of the cash-in transaction is unknown. This status is also set if the *lpNoteNumberList* details are not known or are not reliable.
WFS_CIM_CIRESET	The cash-in transaction ended with a WFS_CMD_CIM_RESET command call, or a reset command call on a compound device class.

usNumOfRefused
Specifies the number of items refused during the currently active or most recently ended cash-in transaction period.

lpNoteNumberList
List of banknote types that were inserted, identified and accepted during the currently active or most recently ended cash-in transaction period. The WFSCIMNOTENUMBER.*ulCount* value within this structure is the count of items of identified and accepted notes during the cash-in transaction period. If items have been rolled back (*wStatus* is WFS_CIM_CIROLLBACK) they will be included in this list. If *wStatus* is WFS_CIM_CIRETRACT or WFS_CIM_CIRESET then identified and accepted items moved to Cash-In or Recycle cash units are included in this list, but items moved to the Retract or Reject cash units are not included. For a description of the WFSCIMNOTENUMBERLIST structure see the definition of the command WFS_INF_CIM_CASH_UNIT_INFO.

~~If a note handling standard is supported then~~ *lpNoteNumberList* includes any level 2 or level 3 notes, and all level 4 fit and unfit notes.

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

lpUnfitNoteNumberList
List of level 4 unfit banknote types that were inserted, identified and accepted during the currently active or most recently ended cash-in transaction period. The WFSCIMNOTENUMBER.*ulCount* value within this structure is the count of items of identified and accepted level 4 unfit notes during the cash-in transaction period. If items have been rolled back (*wStatus* is WFS_CIM_CIROLLBACK) they will be included in this list. If *wStatus* is WFS_CIM_CIRETRACT or WFS_CIM_CIRESET then identified and accepted items moved to Cash-In units are included in this list, but items moved to the Retract or Reject cash units are not included. For a description of the WFSCIMNOTENUMBERLIST structure see the definition of the command WFS_INF_CIM_CASH_UNIT_INFO.

lpUnfitNoteNumberList is a subset of *lpNoteNumberList* where all the accepted notes are listed.

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

Comments None.

5.8 WFS_INF_CIM_GET_P6_INFO

Description This command is used to get information about the number of level 2 / level 3 notes detected and the number of level 2 / level 3 signatures created. The level 2 / level 3 information is available from the point where the WFS_EXEE_CIM_INPUT_P6 (or WFS_EXEE_CDM_INPUT_P6) event is generated until one of the following CIM commands is executed:

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

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

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

Input Param None.

Output Param LPWFSCIMP6INFO *lppP6Info;

Pointer to a NULL-terminated array of pointers to WFSCIMP6INFO structures, one structure for every level:

```
typedef struct _wfs_cim_P6_Info
     {
     USHORT                     usLevel;
     LPWFSCIMNOTENUMBERLIST     lpNoteNumberList;
     USHORT                     usNumOfSignatures;
     } WFSCIMP6INFO, *LPWFSCIMP6INFO;
```

usLevel
Defines the note level. Possible values are:

Value	Meaning
WFS_CIM_LEVEL_2	Information for level 2 notes.
WFS_CIM_LEVEL_3	Information for level 3 notes.

lpNoteNumberList
List of banknote types that were recognized as level 2 or level 3 notes. The WFSCIMNOTENUMBER.*ulCount* values are the count of level 2 or level 3 notes. If the pointer is NULL, no level 2 or level 3 notes were recognized. For a description of the WFSCIMNOTENUMBERLIST structure see the definition of the command WFS_INF_CIM_CASH_UNIT_INFO.

usNumOfSignatures
Number of level 2 or level 3 signatures of this cash-in transaction. If it is zero no signatures are available.

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

~~**Comments** None.~~

Comments Note: Although this command can be used to get information about level 2 /level 3 notes, the information that it provides is limited. The more recent WFS_INF_CIM_GET_ITEM_INFO and WFS_INF_CIM_GET_ALL_ITEMS_INFO commands provide much more information. It is therefore recommended for future development that WFS_INF_CIM_GET_ITEM_INFO and WFS_INF_CIM_GET_ALL_ITEMS_INFO should be used in preference to this command in order to support the greatest functionality, and this command supported where backwards compatibility is necessary..

5.9 WFS_INF_CIM_GET_P6_SIGNATURE

Description This command is used to get one specific signature. Signatures are available from the point where the WFS_EXEE_CIM_INPUT_P6 (or WFS_EXEE_CDM_INPUT_P6) event is generated until one of the following CIM commands is executed:

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

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

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

This command is used to retrieve the required information on an individual item basis. Applications should loop retrieving the information for each index and for each level reported with the WFS_INF_CIM_GET_P6_INFO command.

Input Param LPWFSCIMGETP6SIGNATURE lpGetP6Signature;

```
typedef struct _wfs_cim_get_P6_signature
     {
     USHORT                     usLevel;
     USHORT                     usIndex;
     } WFSCIMGETP6SIGNATURE, *LPWFSCIMGETP6SIGNATURE;
```

usLevel
Defines the level of the wanted signature. Possible values are:

Value	Meaning
WFS_CIM_LEVEL_2	The application wants a level 2 signature.
WFS_CIM_LEVEL_3	The application wants a level 3 signature.

usIndex
Specifies the index (zero to *usNumOfSignatures*-1) of the required signature.

Note: Signatures may be returned in any order; there is no implied relationship between this index and the order in which items are reported in the *lpNoteNumberList* in WFS_INF_CIM_GET_P6_INFO.

Output Param LPWFSCIMP6SIGNATURE lpP6Signature;

```
typedef struct _wfs_cim_P6_signature
     {
     USHORT                     usNoteId;
     ULONG                      ulLength;
     DWORD                      dwOrientation;
     LPVOID                     lpSignature;
     } WFSCIMP6SIGNATURE, *LPWFSCIMP6SIGNATURE;
```

usNoteId
Identification of note type.

ulLength
Length of the signature in bytes.

dwOrientation
Orientation of the entered banknote. Specified as one of the following flags:

Value	Meaning
WFS_CIM_ORFRONTTOP	If note is inserted wide side as the leading edge, the note was inserted with the front image facing up and the top edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the front image face up and the left edge was inserted first.
WFS_CIM_ORFRONTBOTTOM	If note is inserted wide side as the leading edge, the note was inserted with the front image facing up and the bottom edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the front image face up and the right edge was inserted first.
WFS_CIM_ORBACKTOP	If note is inserted wide side as the leading edge, the note was inserted with the back image facing up and the top edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the back image face up and the left edge was inserted first.
WFS_CIM_ORBACKBOTTOM	If note is inserted wide side as the leading edge, the note was inserted with the back image facing up and the bottom edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the back image face up and the right edge was inserted first.
WFS_CIM_ORUNKNOWN	The orientation for the inserted note can not be determined.
WFS_CIM_ORNOTSUPPORTED	The hardware is not capable to determine the orientation.

lpSignature
Pointer to the returned signature.

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

Comments The application has to call this command multiple in a loop to get all signatures.

Note: Although this command can be used to get information about level 2 /level 3 notes, the information that it provides is limited. The more recent WFS_INF_CIM_GET_ITEM_INFO and WFS_INF_CIM_GET_ALL_ITEMS_INFO commands provide much more information. It is therefore recommended for future development that WFS_INF_CIM_GET_ITEM_INFO and WFS_INF_CIM_GET_ALL_ITEMS_INFO should be used in preference to this command in order to support the greatest functionality, and this command supported where backwards compatibility is necessary.

5.10 WFS_INF_CIM_GET_ITEM_INFO

Description This command is used to get information about ~~the number of level 1 / level 2 / level 3 / level 4 notes~~a single detected ~~and the number of level 2 / level 3 / level 4 signatures created~~item. This information is available from the point where the first WFS_EXEE_CIM_INFO_AVAILABLE event is generated until one of the following CIM commands is executed:

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

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

WFS_CMD_CDM_DISPENSE, WFS_CMD_CDM_COUNT, WFS_CMD_CDM_PRESENT, WFS_CMD_CDM_RETRACT, WFS_CMD_CDM_REJECT, WFS_CMD_CDM_OPEN_SHUTTER, WFS_CMD_CDM_CLOSE_SHUTTER, WFS_CMD_CDM_RESET, WFS_CMD_CDM_START_EXCHANGE, WFS_CMD_CDM_END_EXCHANGE, WFS_CMD_CDM_CALIBRATE_CASH_UNIT, WFS_CMD_CDM_TEST_CASH_UNITS. This command is similar to the WFS_INF_CIM_GET_P6_SIGNATURE command but returns additional information for level 2 / level 3 notes and also returns information relating to level 4 notes. The WFS_INF_CIM_GET_P6_INFO command, the WFS_INF_CIM_GET_P6_SIGNATURE command and the WFS_EXEE_CIM_INPUT_P6 event only relate to level 2 and level 3 notes. The WFS_EXEE_CIM_INPUT_P6 event signals that a suspected forgery has been detected and is only generated when level 2 and/or level 3 notes are detected.

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

Input Param LPWFSCIMGETITEMINFO lpGetItemInfo;

```
typedef struct _wfs_cim_get_item_info
     {
     USHORT                     usLevel;
     USHORT                     usIndex;
     DWORD                      dwItemInfoType;
     } WFSCIMGETITEMINFO, *LPWFSCIMGETITEMINFO;
```

usLevel
Defines the note level. Possible values are:

Value	Meaning
WFS_CIM_LEVEL_1	Information for a level 1 note is required. Only an image file can be retrieved for level 1 notes.
WFS_CIM_LEVEL_2	Information for a level 2 note is required. On systems that do not ~~support note handling standards~~classify notes as level 2 this value cannot be used and WFS_ERR_INVALID_DATA will be returned.
WFS_CIM_LEVEL_3	Information for a level 3 note is required. On systems that do not ~~support note handling standards~~classify notes as level 3 this value cannot be used and WFS_ERR_INVALID_DATA will be returned.
WFS_CIM_LEVEL_4	Information for a level 4 note is required. ~~This value is also used to retrieve item information on systems that do not support note handling standards.~~

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

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

Value	Meaning
WFS_CIM_ITEM_SERIALNUMBER	Serial Number of the item.
WFS_CIM_ITEM_SIGNATURE	Signature of the item.
WFS_CIM_ITEM_IMAGEFILE	Image file of the item.

Output Param LPWFSCIMITEMINFO lpItemInfo;

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

```
typedef struct _wfs_cim_item_info
     {
     USHORT                    usNoteID;
     LPWSTR                    lpszSerialNumber;
     LPWFSCIMP6SIGNATURE       lpP6Signature;
     LPSTR                     lpszImageFileName;
     } WFSCIMITEMINFO, *LPWFSCIMITEMINFO;
```

usNoteID
Identification of note type. This value will be zero for level 1 items.

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

lpP6Signature
This field contains the signature for the item, see the WFS_INF_CIM_GET_P6_SIGNATURE command for a description of the contents. If no signature is available or has not been requested then this field is NULL.

lpszImageFileName
Full file path to an image file containing the serial number(s). The format for the file is vendor and/or device specific. The file extension (if any) may be used to determine its format. If the Service Provider does not support this function or the image file has not been requested then *lpszImageFileName* is NULL. The format for the file is vendor and/or device specific. The file extension (if any) may be used to determine its format. The application is responsible for the use and management of this file. For example, the application can transfer the image files to a directory which is managed by the application.

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

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

5.11 WFS_INF_CIM_POSITION_CAPABILITIES

Description This command allows the application to get additional information about the use assigned to each position available in the device.

Input Param None.

Output Param LPWFSCIMPOSCAPABILITIES lpPosCaps;

```
typedef struct _wfs_cim_pos_capabilities
     {
     LPWFSCIMPOSCAPS             *lppPosCapabilities;
     } WFSCIMPOSCAPABILITIES, *LPWFSCIMPOSCAPABILITIES;
```

lppPosCapabilities
Pointer to a NULL-terminated array of pointers to WFSCIMPOSCAPS structures. There is one structure for each position configured in the Service Provider.

```
typedef struct _wfs_cim_pos_caps
     {
     WORD                      fwPosition;
     WORD                      fwUsage;
     BOOL                      bShutterControl;
     BOOL                      bItemsTakenSensor;
     BOOL                      bItemsInsertedSensor;
     WORD                      fwRetractAreas;
     LPSTR                     lpszExtra;
     BOOL                      bPresentControl;
     BOOL                      bPreparePresent;
     } WFSCIMPOSCAPS, *LPWFSCIMPOSCAPS;
```

fwPosition
Specifies one of the CIM input or output positions as one of the following values:

Value	Meaning
WFS_CIM_POSINLEFT	Left input position.
WFS_CIM_POSINRIGHT	Right input position.
WFS_CIM_POSINCENTER	Center input position.
WFS_CIM_POSINTOP	Top input position.
WFS_CIM_POSINBOTTOM	Bottom input position.
WFS_CIM_POSINFRONT	Front input position.
WFS_CIM_POSINREAR	Rear input position.
WFS_CIM_POSOUTLEFT	Left output position.
WFS_CIM_POSOUTRIGHT	Right output position.
WFS_CIM_POSOUTCENTER	Center output position.
WFS_CIM_POSOUTTOP	Top output position.
WFS_CIM_POSOUTBOTTOM	Bottom output position.
WFS_CIM_POSOUTFRONT	Front output position.
WFS_CIM_POSOUTREAR	Rear output position.

fwUsage
Indicates if an output position is used to reject or rollback as a combination of the following flags:

Value	Meaning
WFS_CIM_POSIN	It is an input position.
WFS_CIM_POSREFUSE	It is an output position used to refuse items.
WFS_CIM_POSROLLBACK	It is an output position used to rollback items.

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. In either case the WFS_CMD_CIM_PRESENT_MEDIA command may be used if the *bPresentControl* field is reported as FALSE. The *bShutterControl* field is always set to TRUE if the described position has no shutter.

bItemsTakenSensor
Specifies whether or not the described position 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_ITEMSTAKEN event. If set to FALSE this event is not generated. This field relates to output and refused positions.

bItemsInsertedSensor
Specifies whether the described position 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.

fwRetractAreas
Specifies the areas to which items may be retracted from this position. If the device does not have a retract capability this field will be WFS_CIM_RA_NOTSUPP. Otherwise this field will be set to a combination of the following flags:

Value	Meaning
WFS_CIM_RA_RETRACT	Items may be retracted to a retract cash unit.
WFS_CIM_RA_REJECT	Items may be retracted to a reject cash unit.
WFS_CIM_RA_TRANSPORT	Items may be retracted to the transport.
WFS_CIM_RA_STACKER	Items may be retracted to the intermediate stacker.
WFS_CIM_RA_BILLCASSETTES	Items may be retracted to item cassettes, i.e. cash-in and recycle cash units.
WFS_CIM_RA_CASHIN	Items may be retracted to a cash-in cash unit.

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

bPresentControl
Specifies how the presenting of media items is controlled. If *bPresentControl* is TRUE then the WFS_CMD_CIM_PRESENT_MEDIA command is not supported and items are moved to the output position for removal as part of the relevant command, e.g. WFS_CMD_CIM_CASH_IN or WFS_CMD_CIM_CASH_IN_ROLLBACK where there is implicit shutter control. If *bPresentControl* is FALSE then items returned or rejected can be moved to the output position using the WFS_CMD_CIM_PRESENT_MEDIA command, this includes items returned or rejected as part of a WFS_CMD_CIM_CASH_IN or WFS_CMD_CIM_CASH_IN_ROLLBACK operation. The WFS_CMD_CIM_PRESENT_MEDIA command will open and close the shutter implicitly.

bPreparePresent
Specifies how the presenting of items is controlled. If *bPreparePresent* is FALSE then items to be removed are moved to the output position as part of the relevant command e.g. WFS_CMD_CIM_OPEN_SHUTTER or WFS_CMD_CIM_PRESENT_MEDIA or WFS_CMD_CIM_CASH_IN_ROLLBACK. If *bPreparePresent* is TRUE then items are moved to the output position using the WFS_CMD_CIM_PREPARE_PRESENT command.

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

Comments None.

5.12 WFS_INF_CIM_REPLENISH_TARGET

Description This command is used to determine which cash units can be specified as target cash units for a given source cash unit with the WFS_CMD_CIM_REPLENISH command. For example it can be used to determine which targets can be used for replenishment from a replenishment container or from a recycle cash unit.

Input Param LPWFSCIMREPINFO lpReplenishInfo;

```
typedef struct _wfs_cim_replenish_info
      {
      USHORT                    usNumberSource;
      } WFSCIMREPINFO, *LPWFSCIMREPINFO;
```

usNumberSource
Index number of the logical cash unit which would be used as the source of the replenishment operation. This is the index number identifier defined in the *usNumber* field of the WFSCIMCASHIN structure of the output data of the WFS_INF_CIM_CASH_UNIT_INFO command.

Output Param LPWFSCIMREPINFORES lpReplenishInfoResult;

```
typedef struct _wfs_cim_replenish_info_result
      {
      LPWFSCIMREPINFOTARGET           *lppReplenishTargets;
      } WFSCIMREPINFORES, *LPWFSCIMREPINFORES;
```

lppReplenishTargets
Pointer to a NULL-terminated array of pointers to WFSCIMREPINFOTARGET structures. This output parameter will be NULL if no suitable target was found:

```
typedef struct_wfs_cim_replenish_info_target
      {
      USHORT                    usNumberTarget;
      } WFSCIMREPINFOTARGET, *LPWFSCIMREPINFOTARGET;
```

usNumberTarget
Index number of the logical cash unit that can be used as a target. This is the index number identifier defined in the *usNumber* field of the WFSCIMCASHIN structure of the output data of the WFS_INF_CIM_CASH_UNIT_INFO command.

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

Comments None.

5.13 WFS_INF_CIM_DEVICELOCK_STATUS

Description This command is used to retrieve the lock/unlock statuses of the CIM device and each of its cash units. If the physical lock/unlock of both the CIM device and the cash units are not supported then the WFS_ERR_UNSUPP_CATEGORY error will be returned.

Input Param None.

Output Param LPWFSCIMDEVICELOCKSTATUS lpDevLockStatus;

```
typedef struct _wfs_cim_device_lock_status
     {
     WORD                         wDeviceLockStatus;
     LPWFSCIMCASHUNITLOCK         *lppCashUnitLock;
     } WFSCIMDEVICELOCKSTATUS, *LPWFSCIMDEVICELOCKSTATUS;
```

wDeviceLockStatus
Specifies the physical lock/unlock status of the CIM device:

Value	Meaning
WFS_CIM_LOCK	The device is physically locked.
WFS_CIM_UNLOCK	The device is physically unlocked.
WFS_CIM_LOCKUNKNOWN	Due to a hardware error or other condition, the physical lock/unlock status of the device cannot be determined.
WFS_CIM_LOCKNOTSUPPORTED	The Service Provider does not support physical lock/unlock control of the device.

lppCashUnitLock
Pointer to a NULL-terminated array of pointers to WFSCIMCASHUNITLOCK structures, which specifies the physical lock/unlock status of cash units. Cash units that do not support the physical lock/unlock control are not contained in the array. If there are no cash units that support physical lock/unlock control this will be a NULL pointer.

```
typedef struct _wfs_cim_cash_unit_lock
     {
     LPSTR                        lpPhysicalPositionName;
     WORD                         wCashUnitLockStatus;
     } WFSCIMCASHUNITLOCK, *LPWFSCIMCASHUNITLOCK;
```

lpPhysicalPositionName
A name identifying the physical location of the cash unit within the CIM. This name is the same as the *lpPhysicalPositionName* in the WFSCIMPHCU structure of the WFS_INF_CIM_CASH_UNIT_INFO command.

wCashUnitLockStatus
Specifies the physical lock/unlock status of cash units supported, as one of the following values:

Value	Meaning
WFS_CIM_LOCK	The cash unit is physically locked.
WFS_CIM_UNLOCK	The cash unit is physically unlocked.
WFS_CIM_LOCKUNKNOWN	Due to a hardware error or other condition, the physical lock/unlock status of the cash unit cannot be determined.

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

Comments None.

5.14 WFS_INF_CIM_CASH_UNIT_CAPABILITIES

Description This command is used to retrieve information on cash unit capabilities. It does not provide information on status or counters of cash units.

This command can be seen as an extension to the WFS_INF_CIM_CASH_UNIT_INFO command as it will always result in the same contents with regard to *usNumber* and the physical cash unit information.

Input Param None.

Output Param LPWFSCIMCASHCAPABILITIES lpCashCaps;

```
typedef struct _wfs_cim_cash_caps
     {
     USHORT                         usCount;
     LPWFSCIMCASHUNITCAPABILITIES   *lppCashUnitCaps;
     } WFSCIMCASHCAPABILITIES, *LPWFSCIMCASHCAPABILITIES;
```

usCount
Number of WFSCIMCASHUNITCAPABILITIES structures returned in *lppCashUnitCaps*.

lppCashUnitCaps
Pointer to an array of pointers to WFSCIMCASHUNITCAPABILITIES structures:

```
typedef struct _wfs_cim_cash_unit_capabilities
     {
     USHORT                    usNumber;
     USHORT                    usNumPhysicalCUs;
     LPWFSCIMPHCUCAPABILITIES  *lppPhysical;
     BOOL                      bRetractNoteCountThresholds;
     LPSTR                     lpszExtra;
     DWORD                     fwPossibleItemTypes;
     } WFSCIMCASHUNITCAPABILITIES, *LPWFSCIMCASHUNITCAPABILITIES;
```

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.

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 WFSCIMPHCUCAPABILITIES structures:

```
typedef struct _wfs_cim_physicalcu_capabilities
     {
     LPSTR                    lpPhysicalPositionName;
     ULONG                    ulMaximum;
     BOOL                     bHardwareSensors;
     LPSTR                    lpszExtra;
     } WFSCIMPHCUCAPABILITIES, *LPWFSCIMPHCUCAPABILITIES;
```

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 or IPM to identify shared cash units/media bins.

ulMaximum
Maximum count of items in the physical cash unit. No threshold event will be generated when this value is reached. This value is persistent.

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

bRetractNoteCountThresholds
This field is only valid for cash units of type WFS_CIM_TYPERETRACTCASSETTE. It specifies whether the CIM retract cassette capacity is based on the number of notes, and therefore whether threshold events are generated based on note counts or the number of retract operations. If this value is set to TRUE, threshold events for retract cassettes are generated based on the number of notes, when *ulCashInCount* reaches the *ulMaximum* value. If this value is set to FALSE, threshold events for retract cassettes are generated based on the number of retract operations, when *ulCount* reaches the *ulMaximum* value.

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

fwPossibleItemTypes
Specifies the type of items the cash unit can be configured to accept as a combination of flags. The flags are defined as the same values listed in the *fwItemType* field of the WFSCIMCASHIN structure (see section 5.3). The WFS_INF_CIM_CASH_UNIT_INFO command describes the item types currently configured for a cash unit. This field provides the possible item types values that can be configured for a cash unit using the WFS_CMD_CIM_CONFIGURE_CASH_IN_UNITS command.

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

Comments None.

5.15 WFS_INF_CIM_DEPLETE_SOURCE

Description This command is used to determine which cash units can be specified as source cash units for a given target cash unit with the WFS_CMD_CIM_DEPLETE command. For example it can be used to determine which sources can be used for depletion to a replenishment container or to a cash-in cash unit.

Input Param LPWFSCIMDEPINFO lpDepleteInfo;

```
typedef struct _wfs_cim_deplete_info
     {
     USHORT                    usNumberTarget;
     } WFSCIMDEPINFO, *LPWFSCIMDEPINFO;
```

usNumberTarget
Index number of the logical cash unit which would be used as the target of the depletion operation. This is the index number identifier defined in the *usNumber* field of the WFSCIMCASHIN structure of the output data of the WFS_INF_CIM_CASH_UNIT_INFO command.

Output Param LPWFSCIMDEPINFORES lpDepleteInfoResult;

```
typedef struct _wfs_cim_deplete_info_result
     {
     LPWFSCIMDEPINFOSOURCE          *lppDepleteSources;
     } WFSCIMDEPINFORES, *LPWFSCIMDEPINFORES;
```

lppDepleteSources
Pointer to a NULL-terminated array of pointers to WFSCIMDEPINFOSOURCE structures. This output parameter will be NULL if no suitable source was found:

```
typedef struct_wfs_cim_deplete_info_source
     {
     USHORT                    usNumberSource;
     } WFSCIMDEPINFOSOURCE, *LPWFSCIMDEPINFOSOURCE;
```

usNumberSource
Index number of the logical cash unit that can be used as a source. This is the index number identifier defined in the *usNumber* field of the WFSCIMCASHIN structure of the output data of the WFS_INF_CIM_CASH_UNIT_INFO command.

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

Comments None.

5.16 WFS_INF_CIM_GET_ALL_ITEMS_INFO

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

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

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

WFS_CMD_CDM_DISPENSE, WFS_CMD_CDM_COUNT, WFS_CMD_CDM_PRESENT, WFS_CMD_CDM_RETRACT, WFS_CMD_CDM_REJECT, WFS_CMD_CDM_OPEN_SHUTTER, WFS_CMD_CDM_CLOSE_SHUTTER, WFS_CMD_CDM_RESET, WFS_CMD_CDM_START_EXCHANGE, WFS_CMD_CDM_END_EXCHANGE, WFS_CMD_CDM_CALIBRATE_CASH_UNIT, WFS_CMD_CDM_TEST_CASH_UNITS. This command is similar to the WFS_INF_CIM_GET_P6_SIGNATURE command but returns additional information for level 2 / level 3 notes and also returns information relating to level 4 notes. The WFS_INF_CIM_GET_P6_INFO command, the WFS_INF_CIM_GET_P6_SIGNATURE command and the WFS_EXEE_CIM_INPUT_P6 event only relate to level 2 and level 3 notes. The WFS_EXEE_CIM_INPUT_P6 event ~~signals that a suspected forgery has been detected and~~ is only generated when level 2 and/or level 3 notes are detected.

Input Param LPWFSCIMGETALLITEMSINFO lpGetAllItemsInfo;

```
typedef struct _wfs_cim_get_all_items_info
   {
   USHORT                    usLevel;
   } WFSCIMGETALLITEMSINFO, *LPWFSCIMGETALLITEMSINFO;
```

usLevel
Defines the note level. Possible values are:

Value	Meaning
WFS_CIM_LEVEL_1	Information for a level 1 note is required. Only an image file can be retrieved for level 1 notes.
WFS_CIM_LEVEL_2	Information for level 2 notes is to be returned with the *lpAllItemsInfo* output parameter. On systems that do not classify notes as level 2 this value cannot be used and WFS_ERR_INVALID_DATA will be returned.
WFS_CIM_LEVEL_3	Information for level 3 notes is to be returned with the *lpAllItemsInfo* output parameter. On systems that do not classify notes as level 3 this value cannot be used and WFS_ERR_INVALID_DATA will be returned.
WFS_CIM_LEVEL_4	Information for level 4 notes is to be returned with the *lpAllItemsInfo* output parameter. ~~This value is also used to retrieve item information on systems that do not support note handling standards.~~
WFS_CIM_LEVEL_ALL	Information for all levels ~~all items~~ is to be returned with the *lpAllItemsInfo* output parameter.

Output Param LPWFSCIMALLITEMSINFO lpAllItemsInfo;

```
typedef struct _wfs_cim_all_items_info
     {
     USHORT                    usCount;
     LPWFSCIMITEMINFOALL       *lppItemsList;
     } WFSCIMALLITEMSINFO, *LPWFSCIMALLITEMSINFO;
```

usCount
Number of WFSCIMITEMINFOALL structures returned in *lppItemsList*.

lppItemsList
Pointer to an array of pointers to WFSCIMITEMINFOALL structures:

```
typedef struct _wfs_cim_item_info_all
     {
     USHORT                    usLevel;
     USHORT                    usNoteID;
     LPWSTR                    lpszSerialNumber;
     DWORD                     dwOrientation;
     LPSTR                     lpszP6SignatureFileName;
     LPSTR                     lpszImageFileName;
     WORD                      wOnBlacklist;
     WORD                      wItemLocation;
     USHORT                    usNumber;
     WORD                      wOnClassificationList;
     WORD                      wItemDeviceLocation;
     } WFSCIMITEMINFOALL, *LPWFSCIMITEMINFOALL;
```

usLevel
Defines the note level. Possible values are:

Value	Meaning
WFS_CIM_LEVEL_1	A level 1 banknote.
WFS_CIM_LEVEL_2	A level 2 banknote.
WFS_CIM_LEVEL_3	A level 3 banknote.
WFS_CIM_LEVEL_4	A level 4 banknote.

usNoteID
Identification of note type. This value will be zero for level 1 items.

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

dwOrientation
Orientation of the entered banknote. Specified as one of the following flags:

Value	Meaning
WFS_CIM_ORFRONTTOP	If note is inserted wide side as the leading edge, the note was inserted with the front image facing up and the top edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the front image face up and the left edge was inserted first.
WFS_CIM_ORFRONTBOTTOM	If note is inserted wide side as the leading edge, the note was inserted with the front image facing up and the bottom edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the front image face up and the right edge was inserted first.

WFS_CIM_ORBACKTOP	If note is inserted wide side as the leading edge, the note was inserted with the back image facing up and the top edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the back image face up and the left edge was inserted first.
WFS_CIM_ORBACKBOTTOM	If note is inserted wide side as the leading edge, the note was inserted with the back image facing up and the bottom edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the back image face up and the right edge was inserted first.
WFS_CIM_ORUNKNOWN	The orientation for the inserted note can not be determined.
WFS_CIM_ORNOTSUPPORTED	The hardware is not capable to determine the orientation.

lpszP6SignatureFileName
Full file path to a binary file containing only the vendor specific P6 signature data as returned with the *lpSignature* parameter of the WFSCIMP6SIGNATURE structure. If no P6 signature is available then this field is NULL.

lpszImageFileName
Full file path to an image file containing the serial number(s). The format for the file is vendor and/or device specific. The file extension (if any) may be used to determine its format. If the Service Provider does not support this function or the image file has not been requested then *lpszImageFileName* is NULL. The format for the file is vendor and/or device specific. The file extension (if any) may be used to determine its format. The application is responsible for the use and management of this file. For example, the application can transfer the image files to a directory which is managed by the application.

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

Value	Meaning
WFS_CIM_ONBLACKLIST	The serial number of the items is on the blacklist.
WFS_CIM_NOTONBLACKLIST	The serial number of the items is not on the blacklist.
WFS_CIM_BLACKLISTUNKNOWN	It is unknown if the serial number of the item is on the blacklist.

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

Value	Meaning
WFS_CIM_LOCATION_DEVICE	The item is inside the device in some position other than a cash unit.
WFS_CIM_LOCATION_CASHUNIT	The item is in a cash unit. The logical cash unit number is defined by *usNumber*.
WFS_CIM_LOCATION_CUSTOMER	The item has been returned to the customer.
WFS_CIM_LOCATION_UNKNOWN	The item location is unknown.

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

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

Value	Meaning
WFS_CIM_CLASSIFICATIONLIST_ON	The serial number of the items is on the classification list.
WFS_CIM_CLASSIFICATIONLIST_NOTON	The serial number of the items is not on the classification list.
WFS_CIM_CLASSIFICATIONLIST_UNKNOWN	It is unknown if the serial number of the item is on the classification list.

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

Value	Meaning
WFS_CIM_DEVLOC_STACKER	The item is in the intermediate stacker.
WFS_CIM_DEVLOC_OUTPUT	The item is at the output position. The items have not been in customer access.
WFS_CIM_DEVLOC_TRANSPORT	The item is at another location in the device.
WFS_CIM_DEVLOC_UNKNOWN	The item is in the device but its location is unknown.

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

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

5.17 WFS_INF_CIM_GET_BLACKLIST

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

Input Param None.

Output Param LPWFSCIMBLACKLIST lpBlacklist;

```
typedef struct _wfs_cim_blacklist
     {
     LPWSTR                      lpszVersion;
     USHORT                      usCount;
     LPWFSCIMBLACKLISTELEMENT   *lppBlacklistElements;
     } WFSCIMBLACKLIST, *LPWFSCIMBLACKLIST;
```

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

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

lppBlacklistElements
Pointer to an array of pointers to WFSCIMBLACKLISTELEMENT structures.

```
typedef struct _wfs_cim_blacklist_element
     {
     LPWSTR                      lpszSerialNumber;
     CHAR                        cCurrencyID[3];
     ULONG                       ulValue;
     } WFSCIMBLACKLISTELEMENT, *LPWFSCIMBLACKLISTELEMENT;
```

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

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

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

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

Comments None.

5.18 WFS_INF_CIM_GET_CLASSIFICATION_LIST

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

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

Input Param None.

Output Param LPWFSCIMCLASSIFICATIONLIST lpClassificationList;

```
typedef struct _wfs_cim_classification_list
     {
     LPWSTR                          lpszVersion;
     USHORT                          usCount;
     LPWFSCIMCLASSIFICATIONELEMENT   *lppClassificationElements;
     } WFSCIMCLASSIFICATIONLIST, *LPWFSCIMCLASSIFICATIONLIST;
```

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

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

lppClassificationElements
Pointer to an array of pointers to WFSCIMCLASSIFICATIONELEMENT structures.

```
typedef struct _wfs_cim_classification_element
     {
     LPWSTR                    lpszSerialNumber;
     CHAR                      cCurrencyID[3];
     ULONG                     ulValue;
     USHORT                    usLevel;
     BOOL                      bUnfit;
     } WFSCIMCLASSIFICATIONELEMENT, *LPWFSCIMCLASSIFICATIONELEMENT;
```

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

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

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

usLevel
Specifies the note level. Possible values are:

Value	Meaning
WFS_CIM_LEVEL_1	The element specifies notes to be treated as level 1 notes.
WFS_CIM_LEVEL_2	The element specifies notes to be treated as level 2 notes.
WFS_CIM_LEVEL_3	The element specifies notes to be treated as level 3 notes.
WFS_CIM_LEVEL_4	The element specifies notes to be treated as level 4 notes.

bUnfit
Specifies whether the item is to be treated as unfit for dispensing. Applies only where *usLevel* is WFS_CIM_LEVEL_4.

Error Codes	Only the generic error codes defined in [Ref. 1] can be generated by this command.
Comments	None.

5.19 WFS_INF_CIM_CASH_UNIT_COUNT_STATUS

Description During normal processing it is possible that the *ulCount* of a cash unit can become inaccurate due to a jam, mis-pick or other error situation. In this case the WFS_INF_CIM_CASH_UNIT_COUNT_STATUS command could be used to report which cash units are known to have an inaccurate *ulCount*. The application can then issue a WFS_CMD_CIM_CASH_UNIT_COUNT command for only those cash units if supported. Or alternatively the notes could be manually counted as part of a replenishment operation. This command returns the cash unit count status of all cash units.

Input Param None.

Output Param LPWFSCIMCASHCOUNTSTATUS lpCashCountStatus;

```
typedef struct _wfs_cim_cash_count_status
     {
     USHORT                              usCount;
     LPWFSCIMCASHUNITCOUNTSTATUS        *lppCashUnitStatus;
     } WFSCIMCASHCOUNTSTATUS, *LPWFSCIMCASHCOUNTSTATUS;
```

usCount
Number of WFSCIMCASHUNITCOUNTSTATUS structures returned in *lppCashUnitStatus*. This value is the same as the *usCount* in the WFSCIMCASHINFO structure of the WFS_INF_CIM_CASH_UNIT_INFO command.

lppCashUnitStatus
Pointer to an array of pointers to WFSCIMCASHUNITCOUNTSTATUS structures:

```
typedef struct _wfs_cim_cash_unit_count_status
     {
     USHORT                          usNumber;
     USHORT                          usAccuracy;
     USHORT                          usNumPhysicalCUs;
     LPWFSCIMPHCUCOUNTSTATUS        *lppPhCashUnitStatus;
     LPSTR                           lpszExtra;
     } WFSCIMCASHUNITCOUNTSTATUS, *LPWFSCIMCASHUNITCOUNTSTATUS;
```

usNumber
Index number of the logical cash unit.

usAccuracy
Describes the accuracy of *ulCount* as one of the following values:

Value	Meaning
WFS_CIM_ACCURACYNOTSUPPORTED	The hardware is not capable to determine the accuracy of *ulCount*.
WFS_CIM_COUNTACCURATE	The *ulCount* is expected to be accurate. The notes were previously counted or replenished and there have since been no events that might have introduced inaccuracy. This value will be reported as a result of the following commands: WFS_CMD_CIM_REPLENISH and WFS_CMD_CIM_CASH_UNIT_COUNT.
WFS_CIM_COUNTACCURATESET	The *ulCount* is expected to be accurate. The notes were previously set and there have since been no events that might have introduced inaccuracy.
WFS_CIM_COUNTINACCURATE	The *ulCount* is likely to be inaccurate. A jam, picking fault, or some other event may have resulted in a counting inaccuracy.

WFS_CIM_ACCURACYUNKNOWN	The accuracy of *ulCount* cannot be determined. This may be due to cash unit insertion or some other hardware event.

usNumPhysicalCUs
This value indicates the number of WFSCIMPHCUCOUNTSTATUS structures returned. It must be at least 1.

lppPhCashUnitStatus
Pointer to an array of pointers to WFSCIMPHCUCOUNTSTATUS structures:

```
typedef struct _wfs_cim_phcu_count_status
      {
      LPSTR              lpPhysicalPositionName;
      USHORT             usAccuracy;
      LPSTR              lpszExtra;
      } WFSCIMPHCUCOUNTSTATUS, *LPWFSCIMPHCUCOUNTSTATUS;
```

lpPhysicalPositionName
A name identifying the physical location of the cash unit within the CIM. This field can be used by CIM Service Providers which are compounded with a CDM or IPM to identify shared cash units/media bins.

usAccuracy
Describes the accuracy of *ulCount* of a physical cash unit. See the description in *lppCashUnitStatus*.

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

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

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

Comments None.

5.20 WFS_INF_CIM_PRESENT_STATUS

Description This command is used to obtain the status of the most recent attempt to present or return items to the customer. This information includes the number of items previously moved to the output position and the number of items which have yet to be returned as a result of the following commands.

WFS_CMD_CIM_CASH_IN
WFS_CMD_CIM_CASH_IN_ROLLBACK
WFS_CMD_CIM_PREPARE_PRESENT
WFS_CMD_CIM_PRESENT_MEDIA
WFS_CMD_CIM_OPEN_SHUTTER (In the case of returning multiple bunches)

Input Param None.

Output Param LPWFSCIMPRESENTSTATUS lpPresentStatus;

```
typedef struct _wfs_cim_present_status
     {
     WORD                          fwPosition;
     WORD                          wPresentState;
     WORD                          wAdditionalBunches;
     USHORT                        usBunchesRemaining;
     LPWFSCIMNOTENUMBERLIST        lpReturnedItems;
     LPWFSCIMNOTENUMBERLIST        lpTotalReturnedItems;
     LPWFSCIMNOTENUMBERLIST        lpRemainingItems;
     LPSTR                         lpszExtra;
     } WFSCIMPRESENTSTATUS, *LPWFSCIMPRESENTSTATUS;
```

fwPosition
Specifies the output position as one of the following values:

Value	Meaning
WFS_CIM_POSOUTLEFT	Left output position.
WFS_CIM_POSOUTRIGHT	Right output position.
WFS_CIM_POSOUTCENTER	Center output position.
WFS_CIM_POSOUTTOP	Top output position.
WFS_CIM_POSOUTBOTTOM	Bottom output position.
WFS_CIM_POSOUTFRONT	Front output position.
WFS_CIM_POSOUTREAR	Rear output position.

wPresentState
Supplies the status of the items that were to be presented by the most recent attempt to present or return items to the customer as one of the following values:

Value	Meaning
WFS_CIM_PRESENTED	The items were presented. This status is set as soon as the customer has access to the items.
WFS_CIM_NOTPRESENTED	The customer has not had access to the items.
WFS_CIM_UNKNOWN	It is not known if the customer had access to the items.

wAdditionalBunches
Specifies whether or not additional bunches of items are remaining to be presented as a result of the most recent operation, set to one of the following values:

Value	Meaning
WFS_CIM_ADDBUNCHNONE	No additional bunches remain.
WFS_CIM_ADDBUNCHONEMORE	At least one additional bunch remains.
WFS_CIM_ADDBUNCHUNKNOWN	It is unknown whether additional bunches remain.

usBunchesRemaining
If *wAdditionalBunches* is WFS_CIM_ADDBUNCHONEMORE, specifies the number of additional bunches of items remaining to be presented as a result of the current operation. If the number of additional bunches is at least one, but the precise number is unknown, *usBunchesRemaining* will be WFS_CIM_NUMBERUNKNOWN. For any other value of *wAdditionalBunches*, *usBunchesRemaining* will be zero.

lpReturnedItems
Pointer to a WFSCIMNOTENUMBERLIST structure holding a list of banknote numbers which have been moved to the output position as a result of the most recent operation.

lpTotalReturnedItems
Pointer to a WFSCIMNOTENUMBERLIST structure holding a list of cumulative banknote numbers which have been moved to the output position. This value will be reset when the WFS_CMD_CIM_CASH_IN_START, WFS_CMD_CIM_CASH_IN, WFS_CMD_CIM_CASH_IN_END, WFS_CMD_CIM_RETRACT, WFS_CMD_CIM_RESET or WFS_CMD_CIM_CASH_IN_ROLLBACK command is executed.

lpRemainingItems
Pointer to a WFSCIMNOTENUMBERLIST structure holding a list of banknote numbers on the intermediate stacker or transport which have not been yet moved to the output position.

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

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

Comments None.

6. Execute Commands

6.1 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_CASH_IN_ROLLBACK, WFS_CMD_CIM_CASH_IN_END, WFS_CMD_CIM_RETRACT or WFS_CMD_CIM_RESET command is sent. Where WFSCIMCAPS.bShutterControl == FALSE this command precedes any explicit operation of the shutters.

WFS_CMD_CIM_RETRACT will terminate a transaction. In this case WFS_CMD_CIM_CASH_IN_END, WFS_CMD_CIM_CASH_IN_ROLLBACK and WFS_CMD_CIM_CASH_IN will report WFS_ERR_CIM_NOCASHINACTIVE. If an application wishes to determine where the notes went during a transaction it can execute a WFS_INF_CIM_CASH_UNIT_INFO before and after the transaction and then derive the difference.

A hardware failure during the cash-in transaction does not reset the note number list information; instead the note number list information will include items that could be accepted and identified up to the point of the hardware failure.

Exchange: This command can be used during an Exchange (*fwExchangeType* == WFS_CIM_DEPOSITINTO) to deposit items accepted from the input position. See section 8.16 for an example flow. Note that WFS_ERR_CIM_EXCHANGEACTIVE would not be generated in this case.

Input Param LPWFSCIMCASHINSTART lpCashInStart;

```
typedef struct _wfs_cim_cash_in_start
     {
     USHORT                       usTellerID;
     BOOL                         bUseRecycleUnits;
     WORD                         fwOutputPosition;
     WORD                         fwInputPosition;
     } WFSCIMCASHINSTART, *LPWFSCIMCASHINSTART;
```

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

bUseRecycleUnits
Specifies whether or not the recycle cash units should be used ~~for money~~when items are cashed in ~~during the transaction period~~on a successful WFS_CMD_CIM_CASH_IN_END command. 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 rollback. The position is set to one of the following values:

Value	Meaning
WFS_CIM_POSNULL	The items will be presented to the default configuration.
WFS_CIM_POSOUTLEFT	The items will be presented to the left output position.
WFS_CIM_POSOUTRIGHT	The items will be presented to the right output position.
WFS_CIM_POSOUTCENTER	The items will be presented to the center output position.
WFS_CIM_POSOUTTOP	The items will be presented to the top output position.
WFS_CIM_POSOUTBOTTOM	The items will be presented to the bottom output position.
WFS_CIM_POSOUTFRONT	The items will be presented to the front output position.

WFS_CIM_POSOUTREAR	The items will be presented to the rear output position.

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

Value	Meaning
WFS_CIM_POSNULL	The cash is inserted from the default configuration.
WFS_CIM_POSINLEFT	The cash is inserted from the left input position.
WFS_CIM_POSINRIGHT	The cash is inserted from the right input position.
WFS_CIM_POSINCENTER	The cash is inserted from the center input position.
WFS_CIM_POSINTOP	The cash is inserted from the top input position.
WFS_CIM_POSINBOTTOM	The cash is inserted from the bottom input position.
WFS_CIM_POSINFRONT	The cash is inserted from the front input position.
WFS_CIM_POSINREAR	The cash is inserted from the rear input position.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_INVALIDTELLERID	The teller ID is invalid. This error will never be generated by a Self-Service CIM.
WFS_ERR_CIM_UNSUPPOSITION	The position specified is not supported.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in the exchange state.
WFS_ERR_CIM_CASHINACTIVE	The CIM is already in the cash-in state due to a previous WFS_CMD_CIM_CASH_IN_START command.
WFS_ERR_CIM_SAFEDOOROPEN	The safe door is open. This device requires the safe door to be closed in order to perform a WFS_CMD_CIM_CASH_IN_START command.

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

Comments None.

6.2 WFS_CMD_CIM_CASH_IN

Description

This command moves items into the CIM from an input position.

On devices with implicit shutter control, the WFS_EXEE_CIM_INSERTITEMS event will be generated when the device is ready to start accepting media.

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 one or more WFS_EXEE_CIM_INPUTREFUSE events will be sent to report the rejection. See also paragraph below regarding returning refused items.

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 level 4 items to be moved there after validation. Level 2 and level 3 items may also be moved to the cash-in stacker, but some devices may immediately move them to a designated cash unit. Items on the stacker will remain there until the current cash-in transaction is either cancelled by the WFS_CMD_CIM_CASH_IN_ROLLBACK command or confirmed by the WFS_CMD_CIM_CASH_IN_END command. These commands will cause any level 2 or level 3 items on the cash-in stacker to be moved to the appropriate cash unit. If there is no cash-in stacker then this command will move items directly to the cash units and the WFS_CMD_CIM_CASH_IN_ROLLBACK command will not be supported. Cash unit information will be updated accordingly whenever notes are moved to a cash unit during this command.

Note that the *fwAcceptor* status field may change value during a cash-in transaction. If media has been retained to cash units during a cash-in transaction, it may mean that *fwAcceptor* is set to WFS_CIM_ACCCUSTOP, which means subsequent cash-in operations may not be possible. In this case, the subsequent command fails with error code WFS_ERR_CIM_CASHUNITERROR.

The *bShutterControl* field of the WFSCIMCAPS 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, or the WFS_CMD_CIM_PRESENT_MEDIA command. If *bShutterControl* is FALSE then this command does not operate the shutter in any way, the application is responsible for all shutter control. If *bShutterControl* is TRUE this command opens the shutter at the start of the command and closes it once bills are inserted.

The *bPresentControl* field of the WFSCIMPOSCAPS structure returned from the WFS_INF_CIM_POSITION_CAPABILITIES query will determine whether or not it is necessary to call the WFS_CMD_CIM_PRESENT_MEDIA command in order to move items to the output position. If *bPresentControl* is TRUE then all items are moved immediately to the correct output position for removal (a WFS_CMD_CIM_OPEN_SHUTTER command will be needed in the case of explicit shutter control). If *bPresentControl* is FALSE then items are not returned immediately and must be presented to the correct output position for removal using the WFS_CMD_CIM_PRESENT_MEDIA command.

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.

Returning items (single bunch):

If *bShutterControl* is TRUE, and a single bunch of items is returned then this command will complete once the notes have been returned. A WFS_SRVE_CIM_ITEMSPRESENTED event will be generated.

If *bShutterControl* is FALSE, and a single bunch of items is returned then this command will complete without generating a WFS_SRVE_CIM_ITEMSPRESENTED event, instead the WFS_SRVE_CIM_ITEMSPRESENTED event will be generated by the subsequent WFS_CMD_CIM_OPEN_SHUTTER or WFS_CMD_CIM_PRESENT_MEDIA command.

Returning items (multiple bunches):

It is ~~also~~ possible that a device ~~may~~will in certain situations return refused ~~notes~~items in multiple ~~subsequent~~ bunches. In this case, ~~the WFS_CMD_CIM_CASH_IN~~this command will not complete until the final bunch has been presented and after the last WFS_SRVE_CIM_ITEMSPRESENTED event has been generated. For these devices *bShutterControl* and *bPresentControl* fields of the WFSCIMCAPS / WFSCIMPOSCAPS structure returned from the WFS_INF_CIM_CAPABILITIES / WFS_INF_CIM_POSITION_CAPABILITIES query must both be TRUE otherwise it will not be possible to return multiple bunches. Additionally it may be possible to request the completion of this command with WFSCancelAsyncRequest before the final bunch is presented so that after the completion of this command the WFS_CMD_CIM_RETRACT or WFS_CMD_CIM_RESET command can be used to move the remaining bunches, although the ability to do this will be hardware dependent.

~~If *bShutterControl* is TRUE, and a single bunch of notes is refused then the WFS_CMD_CIM_CASH_IN command will complete once the notes have been returned. A WFS_SRVE_CIM_ITEMSPRESENTED event will be generated.~~

~~If *bShutterControl* is FALSE, then the WFS_CMD_CIM_CASH_IN command will complete without generating a WFS_SRVE_CIM_ITEMSPRESENTED event. This will be generated by the Open/Close Shutter commands.~~

~~Note that it is not possible to return multiple bunches of notes if bShutterControl is FALSE.~~

Mixed Media Mode: If the device is operating in Mixed Media mode (WFSCIMSTATUS.*wMixedMode* == WFS_CIM_IPMMIXEDMEDIA) the Service Provider will not perform any operation unless the WFS_CMD_IPM_MEDIA_IN command is called or has already been called on the IPM interface.

Exchange: This command can be used during an Exchange (*fwExchangeType* == WFS_CIM_DEPOSITINTO) to accept items from the input position. See section 8.16 for an example flow. Note that WFS_ERR_CIM_EXCHANGEACTIVE would not be generated in this case.

Input Param None.

Output Param LPWFSCIMNOTENUMBERLIST lpNoteNumberList;

lpNoteNumberList
List of banknote numbers which have been identified and accepted during execution of this command. Refused items are not included in this *lpNoteNumberList* field. If the whole input was refused then this field will be NULL and one or more WFS_EXEE_CIM_INPUTREFUSE events will be generated. If only part of the input was refused then this field will contain the banknote numbers of the accepted items and one or more WFS_EXEE_CIM_INPUTREFUSE events will be generated. For a description of the WFSCIMNOTENUMBERLIST structure see the WFS_INF_CIM_CASH_UNIT_INFO command.

The *lpNoteNumberList* field contains all notes accepted, ~~if a note handling standard is supported then this includes~~including any level 2 or level 3 notes ~~found~~accepted during the cash-in operation.

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

Value	Meaning
WFS_ERR_CIM_CASHUNITERROR	A problem occurred with a cash unit. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details.
WFS_ERR_CIM_TOOMANYITEMS	There were too many items inserted previously. The cash-in stacker is full at the beginning of this command. This may also be reported where a limit specified by WFS_CMD_CIM_SET_CASH_IN_LIMIT has already been reached at the beginning of this command.
WFS_ERR_CIM_NOITEMS	There were no items to cash-in.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.

WFS_ERR_CIM_SHUTTERNOTCLOSED	Shutter failed to close. In the case of explicit shutter control the application should close the shutter first.
WFS_ERR_CIM_NOCASHINACTIVE	There is no cash-in transaction active.
WFS_ERR_CIM_POSITION_NOT_EMPTY	The output position is not empty so a cash-in is not possible.
WFS_ERR_CIM_SAFEDOOROPEN	The safe door is open. This device requires the safe door to be closed in order to perform a WFS_CMD_CIM_CASH_IN command.
WFS_ERR_CIM_FOREIGN_ITEMS_DETECTED	Foreign items have been detected inside the input position.
WFS_ERR_CIM_SHUTTERNOTOPEN	Shutter failed to open.

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

Value	Meaning
WFS_EXEE_CIM_CASHUNITERROR	A problem occurred with a cash unit.
WFS_EXEE_CIM_INPUT_P6	Level 2 and / or level 3 notes are detected.
WFS_EXEE_CIM_INPUTREFUSE	A part or all of the amount of the cash-in order was refused.
WFS_EXEE_CIM_NOTEERROR	An item detection error occurred.
WFS_EXEE_CIM_SUBCASHIN	A cash-in sub-operation has completed. If the cash-in operation has been divided up into a series of sub-operations under hardware control this event is generated each time one of the sub-cash-in operations completes successfully. It may be used for progress reporting.
WFS_SRVE_CIM_ITEMSINSERTED	Items have been inserted into the cash-in position by the user.
WFS_SRVE_CIM_ITEMSTAKEN	The items have been removed by the user. This event is only generated if the *bItemsTakenSensor* field returned in the capabilities information is TRUE.
WFS_SRVE_CIM_ITEMSPRESENTED	Items have been presented to the user to be taken.
WFS_EXEE_CIM_INFO_AVAILABLE	Information is available for items detected during the cash processing operation.
WFS_EXEE_CIM_INSERTITEMS	Device is ready to accept items from the user.
WFS_USRE_CIM_CASHUNITTHRESHOLD	A threshold condition has occurred in one of the cash units.
WFS_SRVE_CIM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.3 WFS_CMD_CIM_CASH_IN_END

Description This command ends a cash-in transaction. If cash items are on the stacker as a result of a WFS_CMD_CIM_CASH_IN command these items are moved to the appropriate cash units.

The cash-in transaction is ended even if this command does not complete successfully.

Mixed Media Mode:

If the device is operating in Mixed Media mode (WFSCIMSTATUS.*wMixedMode* == WFS_CIM_IPMMIXEDMEDIA) non-cash items, e.g. checks may be moved to an output position or media bin specified by the IPM interface. Additionally, the Service Provider will not perform any operation unless the WFS_CMD_IPM_MEDIA_IN_END command is called or has already been called on the IPM. Alternatively, if WFSCIMCAPS.*bMixedDepositAndRollback* is TRUE, then the WFS_CMD_IPM_MEDIA_IN_ROLLBACK command could be used instead of the WFS_CMD_IPM_MEDIA_IN_END command in order to deposit the bills and return the checks.

Where IPM items may be presented the *bPresentControl* field of the WFSCIMPOSCAPS structure returned from the WFS_INF_CIM_POSITION_CAPABILITIES query will determine whether or not it is necessary to call the WFS_CMD_CIM_PRESENT_MEDIA command in order to move items to the output position. If *bPresentControl* is TRUE then all items are moved immediately to the correct output position for removal. If *bPresentControl* is FALSE then items are not returned immediately and must be presented to the correct output position for removal using the WFS_CMD_CIM_PRESENT_MEDIA command.

Exchange: This command can be used during an Exchange (*fwExchangeType* == WFS_CIM_DEPOSITINTO) to deposit items accepted from the input position. See section 8.16 for an example flow. Note that WFS_ERR_CIM_EXCHANGEACTIVE would not be generated in this case.

Input Param None.

Output Param LPWFSCIMCASHINFO lpCashInfo;

lpCashInfo
List of cash units that have taken cash items and the type of cash items they have taken during the current transaction. For a description of the WFSCIMCASHINFO structure see the definition of the WFS_INF_CIM_CASH_UNIT_INFO command. The structure returned only contains data related to the current transaction, e.g. *ulCount* defines the number of banknotes or coins in the cash unit for this transaction.

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

Value	Meaning
WFS_ERR_CIM_CASHUNITERROR	A problem occurred with a cash unit. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details.
WFS_ERR_CIM_NOITEMS	There were no items to cash-in.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.
WFS_ERR_CIM_NOCASHINACTIVE	There is no cash-in transaction active.
WFS_ERR_CIM_POSITION_NOT_EMPTY	The input or output position is not empty.
WFS_ERR_CIM_SAFEDOOROPEN	The safe door is open. This device requires the safe door to be closed in order to perform a WFS_CMD_CIM_CASH_IN_END command.

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

Value	Meaning
WFS_USRE_CIM_CASHUNITTHRESHOLD	A threshold condition has occurred in one of the cash units.
WFS_SRVE_CIM_CASHUNITINFOCHANGED	A cash unit was changed.
WFS_EXEE_CIM_CASHUNITERROR	A problem occurred with the cash unit.

WFS_EXEE_CIM_INPUT_P6	Level 2 and / or level 3 notes are detected during this operation.
WFS_EXEE_CIM_INFO_AVAILABLE	Information is available for items detected during the cash processing operation.
WFS_EXEE_CIM_NOTEERROR	An item detection error occurred.
WFS_SRVE_CIM_ITEMSTAKEN	The items have been removed by the user. This event is only generated during a Mixed Media transaction where the IPM items are presented and taken and the WFSCIMCAPS.*bItemsTakenSensor* field is TRUE.
WFS_SRVE_CIM_ITEMSPRESENTED	Items have been presented to the user to be taken. This event is only generated during a Mixed Media transaction where the IPM items are presented.
WFS_SRVE_CIM_COUNTS_CHANGED	In Mixed Media mode, counters can be changed by the command WFS_CMD_IPM_MEDIA_IN_END.
WFS_SRVE_CIM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments In the special case where all the items inserted by the customer are classified as level 2 and/or level 3 items and the Service Provider is configured to automatically retain these item types then the WFS_CMD_CIM_CASH_IN_END command will complete with WFS_SUCCESS even if the hardware may have already moved the level 2 and/or level 3 items to their respective cash units on the WFS_CMD_CIM_CASH_IN command and there are no items on escrow at the start of the WFS_CMD_CIM_CASH_IN_END command. This allows the location of the notes retained to be reported in the output parameter. If no items are available for cash-in for any other reason then the WFS_ERR_CIM_NOITEMS error code is returned.

6.4 WFS_CMD_CIM_CASH_IN_ROLLBACK

Description This command is used to roll back a cash-in transaction. It causes all the cash items 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 WFSCIMCAPS 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, or WFS_CMD_CIM_PRESENT_MEDIA command. If *bShutterControl* is FALSE then this command does not operate the shutter in any way, the application is responsible for all shutter control. If *bShutterControl* is TRUE then this command opens the shutter and it is closed when all items are removed.

The *bPresentControl* field of the WFSCIMPOSCAPS structure returned from the WFS_INF_CIM_POSITION_CAPABILITIES query will determine whether or not it is necessary to call the WFS_CMD_CIM_PRESENT_MEDIA command in order to move items to the output position. If *bPresentControl* is TRUE then all items are moved immediately to the correct output position for removal (a WFS_CMD_CIM_OPEN_SHUTTER command will be needed in the case of explicit shutter control). If *bPresentControl* is FALSE then items are not returned immediately and must be presented to the correct output position for removal using the WFS_CMD_CIM_PRESENT_MEDIA command.

Items are returned in a single bunch or multiple bunches in the same way as described for the WFS_CMD_CIM_CASH_IN command.

Mixed Media Mode: If the device is operating in Mixed Media mode (WFSCIMSTATUS.*wMixedMode* == WFS_CIM_IPMMIXEDMEDIA) the Service Provider will not perform any operation unless the WFS_CMD_IPM_MEDIA_IN_ROLLBACK command is called or has already been called on the IPM interface. Alternatively, if the WFSCIMCAPS.*bMixedDepositAndRollback* is TRUE, then the WFS_CMD_IPM_MEDIA_IN_END command could be used instead of the WFS_CMD_IPM_MEDIA_IN_ROLLBACK command in order to deposit the checks and return the items.

Exchange: This command can be used during an Exchange (*fwExchangeType* == WFS_CIM_DEPOSITINTO) to return items accepted from the input position. Note that WFS_ERR_CIM_EXCHANGEACTIVE would not be generated in this case.

Input Param None.

Output Param NULL will be returned unless there were level 2 or level 3 notes inserted in the cash-in transaction that are not returned to the customer~~ because a note handling standard is supported~~.

LPWFSCIMCASHINFO lpCashInfo;

lpCashInfo
List of cash units that have taken banknotes and the type of banknotes they have taken. For a description of the WFSCIMCASHINFO structure see the definition of the WFS_INF_CIM_CASH_UNIT_INFO command. The structure returned only contains data related to the current transaction, e.g. *ulCount* defines the number of notes in the cash unit for this transaction.

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

Value	Meaning
WFS_ERR_CIM_CASHUNITERROR	A problem occurred with a cash unit. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details.
WFS_ERR_CIM_SHUTTERNOTOPEN	Shutter failed to open. In the case of explicit shutter control the application may have failed to open the shutter before issuing the command.

WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in the exchange state.
WFS_ERR_CIM_NOCASHINACTIVE	There is no current cash-in transaction.
WFS_ERR_CIM_POSITION_NOT_EMPTY	The input or output position is not empty.
WFS_ERR_CIM_NOITEMS	There were no items to rollback.

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

Value	Meaning
WFS_EXEE_CIM_CASHUNITERROR	A problem occurred with a cash unit.
WFS_SRVE_CIM_ITEMSTAKEN	The items have been removed by the user. This event is only generated if the *bItemsTakenSensor* field returned in the capabilities information is TRUE.
WFS_SRVE_CIM_ITEMSPRESENTED	Items have been presented to the user to be taken.
WFS_EXEE_CIM_INPUT_P6	Level 2 and / or level 3 notes are detected during this operation.
WFS_EXEE_CIM_INFO_AVAILABLE	Information is available for items detected during the cash processing operation.
WFS_USRE_CIM_CASHUNITTHRESHOLD	A threshold condition has occurred in one of the cash units.
WFS_SRVE_CIM_COUNTS_CHANGED	In Mixed Media mode, counters can be changed by WFS_CMD_IPM_MEDIA_IN_END.
WFS_SRVE_CIM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments In the special case where and all the items inserted by the customer are classified as level 2 and/or level 3 items and the Service Provider is configured to automatically retain these item types then the WFS_CMD_CIM_CASH_IN_ROLLBACK command will complete with WFS_SUCCESS even though no items are returned to the customer. This allows the location of the notes retained to be reported in the output parameter. The application can tell if items have been returned or not via the WFS_SRVE_CIM_ITEMSPRESENTED event. This event will be generated before the command completes when items are returned. This event will not be generated if no items are returned. If no items are available to rollback for any other reason then the WFS_ERR_CIM_NOITEMS error code is returned.

6.5 WFS_CMD_CIM_RETRACT

Description This command retracts items from an output position or internal areas within the CIM. Retracted items will be moved to either a retract bin, a reject bin, cash-in/recycle cash units, the transport or an intermediate stacker area. If items from internal areas within the CIM are preventing items at an output position from being retracted then the items from the internal areas will be retracted first. When the items are retracted from an output position the shutter is closed automatically, even if the *bShutterControl* capability is set to FALSE.

This command terminates a running cash-in transaction. The cash-in transaction is terminated even if this command does not complete successfully.

Mixed Media Mode:

If the device is operating in Mixed Media mode (WFSCIMSTATUS.*wMixedMode* == WFS_CIM_IPMMIXEDMEDIA) this command will not perform any operation unless the WFS_CMD_IPM_RETRACT_MEDIA command is called or has already been called on the IPM interface. Where the parameters for this command and the corresponding WFS_CMD_IPM_RETRACT_MEDIA command conflict, for example the device is physically unable to satisfy both commands, the WFS_CMD_CIM_RETRACT input parameters will be used for all items.

Exchange: This command can be used during an Exchange (*fwExchangeType* == WFS_CIM_DEPOSITINTO) to retract items. Note that WFS_ERR_CIM_EXCHANGEACTIVE would not be generated in this case.

Input Param LPWFSCIMRETRACT lpRetract;

```
typedef struct _wfs_cim_retract
     {
     WORD                          fwOutputPosition;
     USHORT                        usRetractArea;
     USHORT                        usIndex;
     } WFSCIMRETRACT, *LPWFSCIMRETRACT;
```

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

Value	Meaning
WFS_CIM_POSNULL	The default configuration information should be used. This value is also used to retract items from internal CIM locations.
WFS_CIM_POSOUTLEFT	Retract items from the left output position.
WFS_CIM_POSOUTRIGHT	Retract items from the right output position.
WFS_CIM_POSOUTCENTER	Retract items from the center output position.
WFS_CIM_POSOUTTOP	Retract items from the top output position.
WFS_CIM_POSOUTBOTTOM	Retract items from the bottom output position.
WFS_CIM_POSOUTFRONT	Retract items from the front output position.
WFS_CIM_POSOUTREAR	Retract items from the rear output position.

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

Value	Meaning
WFS_CIM_RA_RETRACT	Retract the items to a retract cash unit.
WFS_CIM_RA_REJECT	Retract the items to a reject cash unit.
WFS_CIM_RA_TRANSPORT	Retract the items to the transport.
WFS_CIM_RA_STACKER	Retract the items to the intermediate stacker area.
WFS_CIM_RA_BILLCASSETTES	Retract the items to item cassettes, i.e. cash-in and recycle cash units.

WFS_CIM_RA_CASHIN	Retract the items to a cash-in cash unit. The *fwItemType* of the cash-in cash unit defined in WFSCIMCASHINFO must include (WFS_CIM_CITYPALL | WFS_CIM_CITYPUNFIT).

usIndex
If *usRetractArea* is set to WFS_CIM_RA_RETRACT this field defines the position inside the retract cash units into which the cash is to be retracted. *usIndex* starts with a value of one (1) for the first retract position and increments by one for each subsequent position. If there are several logical retract cash units (of type WFS_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.~~

If *usRetractArea* is set to WFS_CIM_RA_CASHIN this field defines the physical cash unit under the WFS_CIM_TYPECASHIN cash units into which the cash is to be retracted. *usIndex* starts with a value of one (1) and would be incremented from the first physical cash unit of the first logical WFS_CIM_TYPECASHIN cash unit to the last physical cash unit of the last logical WFS_CIM_TYPECASHIN cash unit defined in WFSCIMCASHINFO.

If *usRetractArea* is not set to WFS_CIM_RA_RETRACT or WFS_CIM_RA_CASHIN then the value of this field is ignored.

Output Param LPWFSCIMCASHINFO lpCashInfo;

lpCashInfo
List of cash units that have taken banknotes and the type of banknotes they have taken (including level 2 and level 3 notes~~ if a note handling standard is supported and configured).~~). This pointer can be NULL if *usRetractArea* is set to WFS_CIM_RA_TRANSPORT or WFS_CIM_RA_STACKER. For a description of the WFSCIMCASHINFO structure see the definition of the WFS_INF_CIM_CASH_UNIT_INFO command. The structure returned only contains data related to the current transaction, e.g. *ulCount* defines the number of notes in the cash unit for this transaction. Note that *usNoteID* in the NOTENUMBERLIST will be set to zero for level 1 notes retracted.

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

Value	Meaning
WFS_ERR_CIM_CASHUNITERROR	A retract bin caused a problem. A WFS_EXECUTE_EVENT with an id of WFS_EXEE_CIM_CASHUNITERROR will be posted with the details.
WFS_ERR_CIM_NOITEMS	There were no items to retract.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.
WFS_ERR_CIM_SHUTTERNOTCLOSED	The shutter failed to close.
WFS_ERR_CIM_ITEMSTAKEN	Items were present at the output position at the start of the operation, but were removed before the operation was complete - some or all of the items were not retracted.
WFS_ERR_CIM_INVALIDRETRACTPOSITION	The *usIndex* is not supported.
WFS_ERR_CIM_NOTRETRACTAREA	The retract area specified in *usRetractArea* is not supported.
WFS_ERR_CIM_FOREIGN_ITEMS_DETECTED	Foreign items have been detected in the input position.

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

Value	Meaning
WFS_USRE_CIM_CASHUNITTHRESHOLD	A threshold condition has been reached in a ~~retract bin~~cash unit.
WFS_EXEE_CIM_CASHUNITERROR	An error occurred while attempting to retract to a ~~retract bin~~cash unit.
WFS_EXEE_CIM_NOTEERROR	An item detection error occurred.
WFS_EXEE_CIM_INPUT_P6	Level 2 and / or level 3 notes are detected during this operation.
WFS_SRVE_CIM_ITEMSTAKEN	The items have been removed by the user. This event is only generated if the *bItemsTakenSensor* field returned in the capabilities information is TRUE.
WFS_EXEE_CIM_INFO_AVAILABLE	Information is available for items detected during the cash processing operation.
WFS_SRVE_CIM_CASHUNITINFOCHANGED	A cash unit was updated as a result of this command.
WFS_SRVE_CIM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.6 WFS_CMD_CIM_OPEN_SHUTTER

Description This command opens the shutter.

In cases where multiple bunches are to be returned under explicit shutter control and the first bunch has already been presented and taken and the output position is empty, this command moves the next bunch to the output position before opening the shutter – see sections 8.6 and 8.7. This does not apply if the output position is not empty, for example if items had been re-inserted or dropped back into the output position as the shutter closed.

Input Param LPWORD lpfwPosition;

lpfwPosition
Pointer to the position where the 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:

Value	Meaning
WFS_CIM_POSNULL	The default configuration information should be used.
WFS_CIM_POSINLEFT	Open the shutter of the left input position.
WFS_CIM_POSINRIGHT	Open the shutter of the right input position.
WFS_CIM_POSINCENTER	Open the shutter of the center input position.
WFS_CIM_POSINTOP	Open the shutter of the top input position.
WFS_CIM_POSINBOTTOM	Open the shutter of the bottom input position.
WFS_CIM_POSINFRONT	Open the shutter of the front input position.
WFS_CIM_POSINREAR	Open the shutter of the rear input position.
WFS_CIM_POSOUTLEFT	Open the shutter of the left output position.
WFS_CIM_POSOUTRIGHT	Open the shutter of the right output position.
WFS_CIM_POSOUTCENTER	Open the shutter of the center output position.
WFS_CIM_POSOUTTOP	Open the shutter of the top output position.
WFS_CIM_POSOUTBOTTOM	Open the shutter of the bottom output position.
WFS_CIM_POSOUTFRONT	Open the shutter of the front output position.
WFS_CIM_POSOUTREAR	Open the shutter of the rear output position.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_UNSUPPOSITION	The position specified is not supported.
WFS_ERR_CIM_SHUTTERNOTOPEN	Shutter failed to open.
WFS_ERR_CIM_SHUTTEROPEN	Shutter was already open.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state. Note that this would not apply during an Exchange (*fwExchangeType* == WFS_CIM_DEPOSITINTO).
WFS_ERR_CIM_FOREIGN_ITEMS_DETECTED	Foreign items have been detected in the input position.

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

Value	Meaning
WFS_SRVE_CIM_ITEMSTAKEN	The items have been removed by the user. This event is only generated if the *bItemsTakenSensor* field returned in the capabilities information is TRUE.
WFS_SRVE_CIM_ITEMSINSERTED	Items have been inserted by the user.

WFS_SRVE_CIM_SHUTTERSTATUSCHANGED
The shutter status has changed.

Comments None.

6.7 WFS_CMD_CIM_CLOSE_SHUTTER

Description This command closes the shutter.

Input Param LPWORD lpfwPosition;

lpfwPosition
Pointer to the position where the shutter is 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:

Value	Meaning
WFS_CIM_POSNULL	The default configuration information should be used.
WFS_CIM_POSINLEFT	Close the shutter of the left input position.
WFS_CIM_POSINRIGHT	Close the shutter of the right input position.
WFS_CIM_POSINCENTER	Close the shutter of the center input position.
WFS_CIM_POSINTOP	Close the shutter of the top input position.
WFS_CIM_POSINBOTTOM	Close the shutter of the bottom input position.
WFS_CIM_POSINFRONT	Close the shutter of the front input position.
WFS_CIM_POSINREAR	Close the shutter of the rear input position.
WFS_CIM_POSOUTLEFT	Close the shutter of the left output position.
WFS_CIM_POSOUTRIGHT	Close the shutter of the right output position.
WFS_CIM_POSOUTCENTER	Close the shutter of the center output position.
WFS_CIM_POSOUTTOP	Close the shutter of the top output position.
WFS_CIM_POSOUTBOTTOM	Close the shutter of the bottom output position.
WFS_CIM_POSOUTFRONT	Close the shutter of the front output position.
WFS_CIM_POSOUTREAR	Close the shutter of the rear output position.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_UNSUPPOSITION	The position specified is not supported.
WFS_ERR_CIM_SHUTTERCLOSED	Shutter was already closed.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state. Note that this would not apply during an Exchange (*fwExchangeType* == WFS_CIM_DEPOSITINTO).
WFS_ERR_CIM_SHUTTERNOTCLOSED	Shutter failed to close.
WFS_ERR_CIM_TOOMANYITEMS	There were too many items inserted for the shutter to close.
WFS_ERR_CIM_FOREIGN_ITEMS_DETECTED	Foreign items have been detected in the input position. The shutter is open.

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

Value	Meaning
WFS_SRVE_CIM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.8 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 LPWFSCIMTELLERUPDATE lpTellerUpdate;

```
typedef struct _wfs_cim_teller_update
     {
     USHORT                       usAction;
     LPWFSCIMTELLERDETAILS        lpTellerDetails;
     } WFSCIMTELLERUPDATE, *LPWFSCIMTELLERUPDATE;
```

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

Value	Meaning
WFS_CIM_CREATE_TELLER	A teller is to be added.
WFS_CIM_MODIFY_TELLER	Information about an existing teller is to be modified.
WFS_CIM_DELETE_TELLER	A teller is to be removed.

lpTellerDetails
For a specification of the structure 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:

Value	Meaning
WFS_ERR_CIM_INVALIDCURRENCY	The specified currency is not currently available.
WFS_ERR_CIM_INVALIDTELLERID	The teller ID is invalid.
WFS_ERR_CIM_UNSUPPOSITION	The position specified is not supported.
WFS_ERR_CIM_EXCHANGEACTIVE	The target teller is currently in the middle of an exchange operation.

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

Value	Meaning
WFS_SRVE_CIM_TELLERINFOCHANGED	Teller information has been created, modified or deleted.

Comments None.

6.9 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
lpNoteNumberList (contents must be consistent with ulCount)
ulInitialCount
ulDispensedCount
ulPresentedCount
ulRetractedCount
ulRejectCount
ulMinimum

As may the following fields of the WFSCIMPHCU structure:

ulCashInCount
ulCount
ulInitialCount
ulDispensedCount
ulPresentedCount
ulRetractedCount
ulRejectCount

Any other changes must be performed via an exchange operation.

The *lppPhysical* counts must be consistent with the logical cash unit counts. The Service Provider controls whether the logical counts are maintained separately or are based on the sum of the physical counts.

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

The values set by this command are persistent.

Input Param LPWFSCIMCASHINFO lpCUInfo;

The LPWFSCIMCASHINFO structure is specified in the documentation of the WFS_INF_CIM_CASH_UNIT_INFO command. All cash units must be included not just the cash units whose values are to be changed.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_INVALIDCASHUNIT	Invalid cash unit.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.
WFS_ERR_CIM_CASHUNITERROR	A problem occurred with a cash unit. A WFS_EXEE_CIM_CASHUNITERROR event will be posted with the details.

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

Value	Meaning
WFS_USRE_CIM_CASHUNITTHRESHOLD	A threshold condition has been reached in one of the cash units.
WFS_SRVE_CIM_CASHUNITINFOCHANGED	A cash unit was updated as a result of this command.
WFS_EXEE_CIM_CASHUNITERROR	An error occurred while accessing a cash unit.

Comments None.

6.10 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.

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_EXCHANGEACTIVE the CIM is in an exchange state.

While in an exchange state the CIM will process all WFS requests, excluding **WFS[Async]Execute** commands other than WFS_CMD_CIM_END_EXCHANGE and WFS_CMD_CIM_RESET.

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

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

Exchange via CDM and CIM interfaces:

If the exchange is performed separately via the CDM and CIM interfaces then these operations cannot be performed simultaneously. An exchange state must therefore be initiated on each interface in the following sequence:

CDM

(Lock)

WFS_CMD_CDM_START_EXCHANGE

…exchange action…

WFS_CMD_CDM_END_EXCHANGE

(Unlock)

CIM

(Lock)

WFS_CMD_CIM_START_EXCHANGE

…exchange action…

WFS_CMD_CIM_END_EXCHANGE

(Unlock)

In the case of a cash 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. Recycle cash units can be set via either interface. However, if the device has recycle cash units of multiple currencies and/or denominations (or multiple note identifiers associated with the same denomination), then the CIM interface should be used for exchange operations involving these cash units. Those fields which are not common to both the CDM and CIM cash units are left unchanged when an exchange (or WFS_CMD_CDM_SET_CASH_UNIT_INFO or WFS_CMD_CIM_SET_CASH_UNIT_INFO command) is executed on the other interface. For example, if the CDM interface is used to set the current count of notes in the cash unit the CIM *lpNoteNumberList* structure is not changed even if the data becomes inconsistent.

Exchange via the CIM Interface:

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

Input Param LPWFSCIMSTARTEX lpStartEx;

```
typedef struct _wfs_cim_start_ex
     {
     WORD                          fwExchangeType;
     USHORT                        usTellerID;
     USHORT                        usCount;
     LPUSHORT                      lpusCUNumList;
     LPWFSCIMOUTPUT                lpOutput;
     } WFSCIMSTARTEX, *LPWFSCIMSTARTEX;
```

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

Value	Meaning
WFS_CIM_EXBYHAND	The cash units will be replenished manually either by filling or emptying the cash unit by hand or by replacing the cash unit.
WFS_CIM_EXTOCASSETTES	Items will be moved from the replenishment container to the bill cash units. Items will be moved from the bill cash units to the replenishment container. On a cash recycler, the CDM interface should be used to move items from a replenishment container.
WFS_CIM_CLEARRECYCLER	Items will be moved from a recycle cash unit to a cash unit or output position.
WFS_CIM_DEPOSITINTO	Items will be moved from the deposit entrance to the bill cash units. See section 8.16 for an example flow.

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. This is not applicable where *fwExchangeType* is WFS_CIM_DEPOSITINTO as it may not be known in advance which cash units the items will be sorted to.

lpusCUNumList
Pointer to an array of unsigned shorts containing the logical numbers of the cash units to be exchanged.

lpOutput
This field 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 cash unit be emptied.

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

Value	Meaning
WFS_CIM_POSNULL	Move items to a cash unit. If no cash unit is specified in *usNumber*, use the default output position.
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.

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

Output Param LPWFSCIMCASHINFO lpCUInfo;

The WFSCIMCASHINFO 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:

Value	Meaning
WFS_ERR_CIM_INVALIDTELLERID	Invalid teller ID. This error will never be generated by a Self-Service CIM.
WFS_ERR_CIM_CASHUNITERROR	An error occurred with a cash unit while performing the exchange operation. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details.
WFS_ERR_CIM_TOOMANYITEMS	This error is generated if the contents of the recycle cash unit cannot be completely emptied to the output position. The maximum possible number of items is moved to the output position.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is already in an exchange state.
WFS_ERR_CIM_CASHINACTIVE	A cash-in transaction is active.

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

Value	Meaning
WFS_EXEE_CIM_CASHUNITERROR	A cash unit caused an error.
WFS_EXEE_CIM_NOTEERROR	An item detection error occurred.
WFS_USRE_CIM_CASHUNITTHRESHOLD	A threshold condition has occurred in one of the cash units. This event is not generated for recycle cash units.
WFS_SRVE_CIM_CASHUNITINFOCHANGED	A cash unit was changed.
WFS_SRVE_CIM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.11 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~~.~~, including depositing any remaining items where *fwExchangeType* is WFS_CIM_DEPOSITINTO. 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.

The *lppPhysical* counts must be consistent with the logical cash unit counts. The Service Provider controls whether the logical counts are maintained separately or are based on the sum of the physical counts.

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

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

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

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

Input Param LPWFSCIMCASHINFO lpCUInfo;

The LPWFSCIMCASHINFO 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 or the cash units have been replenished mechanically using replenishment or recycling cassettes or where *fwExchangeType* is WFS_CIM_DEPOSITINTO. Otherwise the parameter must contain the complete list of cash unit structures not just the ones that have changed.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_CASHUNITERROR	A cash unit problem occurred that meant no cash units could be updated. One or more WFS_EXEE_CIM_CASHUNITERROR events will be sent with the details.
WFS_ERR_CIM_NOEXCHANGEACTIVE	There is no exchange active.

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

Value	Meaning
WFS_USRE_CIM_CASHUNITTHRESHOLD	A threshold condition has been reached in one of the cash units.
WFS_SRVE_CIM_CASHUNITINFOCHANGED	A cash unit was changed.
WFS_EXEE_CIM_CASHUNITERROR	A cash unit caused an error.

Comments None.

6.12 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.
Error Codes	In addition to the generic error codes defined in [Ref. 1], the following error codes can be generated by this command:

Value	Meaning
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.

Events	Only the generic events defined in [Ref. 1] can be generated by this command.
Comments	None.

6.13 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.

If a cash-in transaction is active, this command will end it (even if this command does not complete successfully). If an exchange state is active then this command will end the exchange state (even if this command does not complete successfully).

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

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

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

The *bShutterControl* field of the WFSCIMCAPS 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, or the WFS_CMD_CIM_PRESENT_MEDIA command. If *bShutterControl* is FALSE then this command does not operate the shutter in any way, the application is responsible for all shutter control. If *bShutterControl* is TRUE then this command operates the shutter as necessary so that the shutter is closed after the command completes successfully and any items returned to the customer have been removed.

The *bPresentControl* field of the WFSCIMPOSCAPS structure returned from the WFS_INF_CIM_POSITION_CAPABILITIES query will determine whether or not it is necessary to call the WFS_CMD_CIM_PRESENT_MEDIA command in order to move items to the output position. If *bPresentControl* is TRUE then all items are moved immediately to the correct output position for removal (a WFS_CMD_CIM_OPEN_SHUTTER command will be needed in the case of explicit shutter control). If *bPresentControl* is FALSE then items are not returned immediately and must be presented to the correct output position for removal using the WFS_CMD_CIM_PRESENT_MEDIA command.

If requested, items are returned in a single bunch or multiple bunches in the same way as described for the WFS_CMD_CIM_CASH_IN command.

Mixed Media Mode:

The value of WFSCIMSTATUS.*wMixedMode* is not changed by this command. Where the items are to be moved to a cash unit, the cash unit must support an *fwItemType* of WFS_CIM_CITYPIPM.

Input Param

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

LPWFSCIMITEMPOSITION lpResetIn;

```
typedef struct _wfs_cim_itemposition
     {
     USHORT                    usNumber;
     LPWFSCIMRETRACT           lpRetractArea;
     WORD                      fwOutputPosition;
     } WFSCIMITEMPOSITION, *LPWFSCIMITEMPOSITION;
```

usNumber
~~In the case of a single cash unit destination~~If non-zero, this value specifies the ~~cash unit~~ *usNumber* (as specified by WFS_INF_CIM_CASH_UNIT_INFO) of the single cash unit to be used for the storage of any items found~~, i.e. when~~.

If items are to be moved to ~~a reject or retract cash unit. In all other cases~~an output position, this value must be zero, ~~i.e. when~~ *lpRetractArea* must be NULL and *fwOutputPosition* specifies where items are to be moved to~~ item cassettes,~~ .

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

lpRetractArea
This field is used if items are to be moved to internal areas of the device, including cash units, the intermediate stacker~~,~~ or the transport~~, a retract cassette or to item cassettes. If items are to be moved to a reject cash unit or to an output position then this field must be NULL.~~. The field is only relevant if *usNumber* is zero. The WFSCIMRETRACT~~, *LPWFSCIMRETRACT;~~ structure is defined in WFS_CMD_CIM_RETRACT.

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

usRetractArea
~~This value specifies~~See the ~~area to which~~description in WFS_CMD_CIM_RETRACT.

usIndex
See the ~~items are to be moved to~~description in ~~. Possible values are:~~

~~Value~~	~~Meaning~~
~~WFS_CIM_RA_RETRACT~~	~~Items will be moved to a retract cash unit. In the case where several cash units of type WFS_CIM_TYPERETRACTCASSETTE exist the *usNumber* field will define which retract unit the items will be moved to.~~
~~WFS_CIM_RA_TRANSPORT~~	~~Items will be moved to the transport.~~
WFS_CMD_CIM_~~RA_STACKER~~	~~Items will be moved to the intermediate stacker area.~~
~~WFS_CIM_RA_BILLCASSETTES~~	~~Items will be moved to item cassettes, i.e. cash-in and recycle cash units.~~

~~*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~~.

fwOutputPosition
The output position to which items are to be moved. ~~If the~~ This field is only used if *usNumber* is ~~non-~~zero ~~or if~~and *lpRetractArea* ~~indicates WFS_CIM_RA_BILLCASSETTES then this field must be zero~~is NULL. The value is set to one of the following values:

Value	Meaning
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.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_CASHUNITERROR	A cash unit caused an error. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details.
WFS_ERR_CIM_UNSUPPOSITION	The position specified is not supported.
WFS_ERR_CIM_INVALIDCASHUNIT	The cash unit number specified is not valid.
WFS_ERR_CIM_INVALIDRETRACTPOSITION	The *usIndex* is not supported.
WFS_ERR_CIM_NOTRETRACTAREA	The retract area specified in *usRetractArea* is not supported.
WFS_ERR_CIM_FOREIGN_ITEMS_DETECTED	Foreign items have been detected in the input position.

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

Value	Meaning
WFS_USRE_CIM_CASHUNITTHRESHOLD	A threshold condition has been reached in one of the cash units.
WFS_EXEE_CIM_CASHUNITERROR	A cash unit caused an error.
WFS_SRVE_CIM_MEDIADETECTED	Media was detected during the reset.
WFS_EXEE_CIM_INPUT_P6	Level 2 and / or level 3 notes are detected during this operation.
WFS_SRVE_CIM_ITEMSTAKEN	The items have been removed by the user. This event is only generated if the *bItemsTakenSensor* field returned in the Capabilities information is TRUE.
WFS_EXEE_CIM_INFO_AVAILABLE	Information is available for items detected during the cash processing operation.
WFS_SRVE_CIM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.14 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~~ unit can take. The *fwPossibleItemTypes* field of the WFSCIMCASHUNITCAPABILITIES structure (see section 5.14) indicates values that can be configured for a given cash unit.

The values set by this command are persistent.

Input Param LPWFSCIMCASHINTYPE *lppCashInType;

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

```
typedef struct _wfs_cim_cash_in_type
      {
      USHORT                     usNumber;
      DWORD                      dwType;
      LPUSHORT                   lpusNoteIDs;
      } WFSCIMCASHINTYPE, *LPWFSCIMCASHINTYPE;
```

usNumber
Logical number of the cash unit.

dwType
Specifies the type of items the cash~~-in~~ unit ~~or recycle unit. Specified~~is to take as a combination of the following flags~~:~~. This modifies the *fwItemType* in a WFSCIMCASHIN (see section 5.3):

Value	Meaning
WFS_CIM_CITYPALL	The cash~~-in~~ unit accepts all fit banknote types These are Level 4 notes which are fit for recycling.
WFS_CIM_CITYPUNFIT	The cash~~-in~~ unit accepts all unfit banknotes. These are level 4 notes which are unfit for recycling.
WFS_CIM_CITYPINDIVIDUAL	The cash~~-in~~ unit or recycle unit accepts all types of fit banknotes specified in the following list.
WFS_CIM_CITYPLEVEL1	Level 1 note types are stored in this cash unit.
WFS_CIM_CITYPLEVEL2	~~If a note handling standard is supported then level~~ Level 2 note types are stored in this cash~~-in~~ unit.
WFS_CIM_CITYPLEVEL3	~~If a note handling standard is supported then level~~ Level 3 note types are stored in this cash~~-in~~ unit.
WFS_CIM_CITYPIPM	The cash~~-in~~ unit can accept items on the IPM interface.
WFS_CIM_CITYPUNFITINDIVIDUAL	The cash unit takes all types of unfit banknotes specified in an individual list. These are level 4 notes which are unfit for recycling. This is only valid when combined with WFS_CIM_CITYPINDIVIDUAL.

See the definition of the WFS_INF_CIM_CASH_UNIT_INFO command for a detailed description.

lpusNoteIDs
Pointer to a zero-terminated list of unsigned shorts which contains the note IDs of the banknotes the cash~~-in cash unit or recycle~~ unit can take. This field only applies if the *dwType* field has the WFS_CIM_CITYPINDIVIDUAL or WFS_CIM_CITYPUNFITINDIVIDUAL flag set.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_INVALIDCASHUNIT	Invalid cash unit. This error will also be created if an invalid logical number of a cash unit is given.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.
WFS_ERR_CIM_CASHUNITNOTEMPTY	The hardware requires that the cash unit is empty before allowing changes.

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

Value	Meaning
WFS_SRVE_CIM_CASHUNITINFOCHANGED	A cash unit was changed.

~~**Comments** None.~~

Comments Using this command it is possible to configure cash units in a highly flexible manner that can satisfy a wide range of requirements.

Example 1: A retract cash unit may be configured to accept Level 2 and 3 notes.

Example 2: A retract cash unit may be configured to take an entire bunch (including Level 1, 2, 3, 4, fit and unfit notes).

It should be noted that the above two use cases are only examples, the combination of which *dwType* values can be configured for any given cash unit will be hardware dependent (see section 5.14).

6.15 WFS_CMD_CIM_CONFIGURE_NOTETYPES

Description This command is used to configure the note types the banknote reader ~~will recognize~~should accept during cash-in. All note types the banknote reader ~~has to recognize~~should accept must be given in the input structure. If an unknown note type is given the error code WFS_ERR_UNSUPP_DATA will be returned.

The values set by this command are persistent.

Input Param LPUSHORT lpusNoteIDs;

lpusNoteIDs
Pointer to a zero-terminated list of unsigned shorts which contains the note IDs of the banknotes 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:

Value	Meaning
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.
WFS_ERR_CIM_CASHINACTIVE	A cash-in transaction is active. This device requires that no cash-in transaction is active in order to perform the command.

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

Comments None.

6.16 WFS_CMD_CIM_CREATE_P6_SIGNATURE

Description This command is used to create a reference signature (normally a level 3 note) that was checked and regarded as a forgery. The reference can be compared with the available signatures of the cash-in transactions to track back the customer.

When this command is executed, the CIM waits for a note to be inserted at the input position, transports the note to the recognition module, creates the signature and then returns the note to the output position.

The *bShutterControl* field of the WFSCIMCAPS 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, or WFS_CMD_CIM_PRESENT_MEDIA command. If *bShutterControl* is FALSE then this command does not operate the shutter in any way, the application is responsible for all shutter control. If *bShutterControl* is TRUE then this command opens and closes the shutter at various times during the command execution and the shutter is finally closed when all items are removed.

The *bPresentControl* field of the WFSCIMPOSCAPS structure returned from the WFS_INF_CIM_POSITION_CAPABILITIES query will determine whether or not it is necessary to call the WFS_CMD_CIM_PRESENT_MEDIA command in order to move items to the output position. If *bPresentControl* is TRUE then all items are moved immediately to the correct output position for removal (a WFS_CMD_CIM_OPEN_SHUTTER command will be needed in the case of explicit shutter control). If *bPresentControl* is FALSE then items are not returned immediately and must be presented to the correct output position for removal using the WFS_CMD_CIM_PRESENT_MEDIA command.

On devices with implicit shutter control, the WFS_EXEE_CIM_INSERTITEMS event will be generated when the device is ready to start accepting media.

The application may have to execute this command repeatedly to make sure that all possible signatures are captured.

If a single note is entered and returned to the customer but cannot be processed fully (e.g. no recognition software in the recognition module, the note is not recognized, etc.) then a WFS_EXEE_CIM_INPUTREFUSE event will be sent and the command will complete with WFS_SUCCESS. In this case, the output parameters will be set as follows, *usNoteId* = zero, *ulLength* = zero, *dwOrientation* = WFS_CIM_ORUNKNOWN and *lpSignature* = NULL.

Input Param None.

Output Param LPWFSCIMP6SIGNATURE lpP6Signature;

```
typedef struct _wfs_cim_P6_signature
     {
     USHORT                     usNoteId;
     ULONG                      ulLength;
     DWORD                      dwOrientation;
     LPVOID                     lpSignature;
     } WFSCIMP6SIGNATURE, *LPWFSCIMP6SIGNATURE;
```

usNoteId
Identification of note type.

ulLength
Length of the signature in bytes.

dwOrientation
Orientation of the entered banknote. Specified as one of the following flags:

Value	Meaning
WFS_CIM_ORFRONTTOP	If note is inserted wide side as the leading edge, the note was inserted with the front image facing up and the top edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the front image face up and the left edge was inserted first.
WFS_CIM_ORFRONTBOTTOM	If note is inserted wide side as the leading edge, the note was inserted with the front image facing up and the bottom edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the front image face up and the right edge was inserted first.
WFS_CIM_ORBACKTOP	If note is inserted wide side as the leading edge, the note was inserted with the back image facing up and the top edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the back image face up and the left edge was inserted first.
WFS_CIM_ORBACKBOTTOM	If note is inserted wide side as the leading edge, the note was inserted with the back image facing up and the bottom edge of the note was inserted first. If the note is inserted short side as the leading edge, the note was inserted with the back image face up and the right edge was inserted first.
WFS_CIM_ORUNKNOWN	The orientation for the inserted note can not be determined.
WFS_CIM_ORNOTSUPPORTED	The hardware is not capable to determine the orientation.

lpSignature
Pointer to the returned signature.

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

Value	Meaning
WFS_ERR_CIM_TOOMANYITEMS	There was more than one banknote inserted for creating a signature.
WFS_ERR_CIM_NOITEMS	There was no banknote to create a signature.
WFS_ERR_CIM_CASHINACTIVE	A cash-in transaction is active.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.
WFS_ERR_CIM_POSITION_NOT_EMPTY	The output position is not empty so a banknote cannot be inserted.
WFS_ERR_CIM_SHUTTERNOTOPEN	Shutter failed to open.
WFS_ERR_CIM_SHUTTERNOTCLOSED	Shutter failed to close.
WFS_ERR_CIM_FOREIGN_ITEMS_DETECTED	Foreign items have been detected in the input position.

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

Value	Meaning
WFS_EXEE_CIM_INPUTREFUSE	The inserted item was no banknote or the note was not recognized.
WFS_SRVE_CIM_ITEMSINSERTED	Items have been inserted into the cash-in position by the user.
WFS_SRVE_CIM_ITEMSTAKEN	Items returned to the user have been taken.

WFS_SRVE_CIM_ITEMSPRESENTED	Items have been presented to the user to be taken.
WFS_EXEE_CIM_NOTEERROR	An item detection error occurred.
WFS_EXEE_CIM_INSERTITEMS	Device is ready to accept items from the user.
WFS_EXEE_CIM_INFO_AVAILABLE	Information is available for items detected during this operation.
WFS_SRVE_CIM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.17 WFS_CMD_CIM_SET_GUIDANCE_LIGHT

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

Input Param LPWFSCIMSETGUIDLIGHT lpSetGuidLight;

```
typedef struct _wfs_cim_set_guidlight
     {
     WORD                      wGuidLight;
     DWORD                     dwCommand;
     } WFSCIMSETGUIDLIGHT, *LPWFSCIMSETGUIDLIGHT;
```

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

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

Value	Meaning	Type
WFS_CIM_GUIDANCE_OFF	The light indicator is turned off.	A
WFS_CIM_GUIDANCE_SLOW_FLASH	The light indicator is set to flash slowly.	B
WFS_CIM_GUIDANCE_MEDIUM_FLASH	The light indicator is set to flash medium frequency.	B
WFS_CIM_GUIDANCE_QUICK_FLASH	The light indicator is set to flash quickly.	B
WFS_CIM_GUIDANCE_CONTINUOUS	The light indicator is turned on continuously (steady).	B
WFS_CIM_GUIDANCE_RED	The light indicator color is set to red.	C
WFS_CIM_GUIDANCE_GREEN	The light indicator color is set to green.	C
WFS_CIM_GUIDANCE_YELLOW	The light indicator color is set to yellow.	C
WFS_CIM_GUIDANCE_BLUE	The light indicator color is set to blue.	C
WFS_CIM_GUIDANCE_CYAN	The light indicator color is set to cyan.	C
WFS_CIM_GUIDANCE_MAGENTA	The light indicator color is set to magenta.	C
WFS_CIM_GUIDANCE_WHITE	The light indicator color is set to white.	C
WFS_CIM_GUIDANCE_ENTRY	The light indicator is set to the entry state.	D
WFS_CIM_GUIDANCE_EXIT	The light indicator is set to the exit state.	D

Output Param None.

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

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

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

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

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

6.18 WFS_CMD_CIM_CONFIGURE_NOTE_READER

Description This command is used to configure the currency description configuration data into the banknote reader module. The format and location of the configuration data is vendor and/or hardware dependent.

Input Param LPWFSCIMCONFIGURENOTEREADER lpConfigureNoteReader;

```
typedef struct _wfs_cim_configure_note_reader
     {
     BOOL                      bLoadAlways;
     } WFSCIMCONFIGURENOTEREADER, *LPWFSCIMCONFIGURENOTEREADER;
```

bLoadAlways
If set to TRUE, the Service Provider loads the currency description data into the note reader, even if it is already loaded.

Output Param LPWFSCIMCONFIGURENOTEREADEROUT lpConfigureNoteReaderOut;

```
typedef struct _wfs_cim_configure_note_reader_out
     {
     BOOL                      bRebootNecessary;
     } WFSCIMCONFIGURENOTEREADEROUT, *LPWFSCIMCONFIGURENOTEREADEROUT;
```

bRebootNecessary
If set to TRUE, the machine needs a reboot before the note reader can be accessed again.

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

Value	Meaning
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.
WFS_ERR_CIM_CASHINACTIVE	A cash-in transaction is active.
WFS_ERR_CIM_LOADFAILED	The load failed because the device is in a state that will not allow the configuration data to be loaded at this time, for example on some devices there may be notes present in the cash units when they should not be.

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

Comments None.

6.19 WFS_CMD_CIM_COMPARE_P6_SIGNATURE

Description This command is used to compare the signatures of a reference banknote with the available signatures of the cash-in transactions.

The reference signatures are created by the WFS_CMD_CIM_CREATE_P6_SIGNATURE command.

The transaction signatures are obtained through the WFS_INF_CIM_GET_P6_SIGNATURE command.

The signatures (1 to 4) of the reference banknote are typically the signatures of the 4 orientations of the banknote.

The WFS_CMD_CIM_COMPARE_P6_SIGNATURE command may return a single indication or a list of indications to the matching signatures, each one associated to a confidence level factor. If the Service Provider does not support the confidence level factor, it returns a single indication to the best matching signature with the confidence level factor set to zero.

If the comparison completed with no matching signatures found then the command returns WFS_SUCCESS with *lppP6SignaturesIndex* set to NULL and *usCount* set to zero.

This command must be used outside of the cash-in transactions and outside of exchange states.

Input Param LPWFSCIMP6COMPARESIGNATURE lpP6CompareSignature;

```
typedef struct _wfs_cim_P6_compare_signature
      {
      LPWFSCIMP6SIGNATURE        *lppP6ReferenceSignatures;
      LPWFSCIMP6SIGNATURE        *lppP6Signatures;
      } WFSCIMP6COMPARESIGNATURE, *LPWFSCIMP6COMPARESIGNATURE;
```

lppP6ReferenceSignatures
Pointer to a NULL-terminated array of pointers to WFSCIMP6SIGNATURE structures.

Each pointer points to the signature corresponding to one orientation of a single reference banknote.

At least one orientation must be provided. If no orientations are provided (this pointer is NULL or points to NULL) the command returns WFS_ERR_INVALID_DATA. For a description of the WFSCIMP6SIGNATURE structure see the definition of the command WFS_CMD_CIM_CREATE_P6_SIGNATURE.

lppP6Signatures
Pointer to a NULL-terminated array of pointers to WFSCIMP6SIGNATURE structures. Each pointer points to a level 2/3 signature, from the cash-in transactions, to be compared with the reference signatures in *lppP6ReferenceSignature.*

At least one signature must be provided. If there are no signatures provided (this pointer is NULL or points to NULL) the command returns WFS_ERR_INVALID_DATA.

For a description of the WFSCIMP6SIGNATURE structure see the definition of the command WFS_INF_CIM_GET_P6_SIGNATURE.

Output Param LPWFSCIMP6COMPARERESULT lpP6CompareResult;

```
typedef struct _wfs_cim_P6_compare_result
      {
      USHORT                      usCount;
      LPWFSCIMP6SIGNATURESINDEX *lppP6SignaturesIndex;
      } WFSCIMP6COMPARERESULT, *LPWFSCIMP6COMPARERESULT;
```

usCount
Number of WFSCIMP6SIGNATURESINDEX structures returned in *lppP6SignaturesIndex*.

lppP6SignaturesIndex
Pointer to a NULL-terminated array of pointers to WFSCIMP6SIGNATURESINDEX structures. This pointer is NULL and *usCount* is zero when the compare operation completes with no match found.

If there are matches found, *lppP6SignaturesIndex* contains the indexes of the matching signatures from the input parameter *lppP6Signatures*.

If there is a match found but the Service Provider does not support the confidence level factor, *lppP6SignaturesIndex* contains a single index with *usConfidenceLevel* set to zero.

```
typedef struct _wfs_cim_P6_signatures_index
     {
     USHORT                    usIndex;
     USHORT                    usConfidenceLevel;
     ULONG                     ulLength;
     LPVOID                    lpComparisonData;
     } WFSCIMP6SIGNATURESINDEX, *LPWFSCIMP6SIGNATURESINDEX;
```

usIndex
Specifies the index (zero to *usNumOfSignatures*-1) of the matching signature from the input parameter *lppP6Signatures.*

usConfidenceLevel
Specifies the level of confidence for the match found. This value is in a scale 1 - 100, where 100 is the maximum confidence level. This value is zero if the Service Provider does not support the confidence level factor.

ulLength
Length of the comparison data in bytes.

lpComparisonData
Pointer to vendor dependent comparison result data. This data may be used as justification for the signature match or confidence level. This pointer is NULL if no additional comparison data is returned.

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

Value	Meaning
WFS_ERR_CIM_CASHINACTIVE	A cash-in transaction is active.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in the exchange state.
WFS_ERR_CIM_INVALIDREFSIG	At least one of the reference signatures is invalid. The application should prompt the operator to carefully retry the creation of the reference signatures.
WFS_ERR_CIM_INVALIDTRNSIG	At least one of the transaction signatures is invalid.

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

Comments Due to the potential for signatures to be large, as well as the possibility that it may be necessary to compare the reference signature with a large number of signatures, applications should be aware of the amount of data passed as input to this command. In some cases, it may be necessary to execute this command more than once, with subsets of the total signatures, and then afterward compare the results from each execution.

6.20 WFS_CMD_CIM_POWER_SAVE_CONTROL

Description This command activates or deactivates the power saving mode.

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

Input Param LPWFSCIMPOWERSAVECONTROL lpPowerSaveControl;

```
typedef struct _wfs_cim_power_save_control
     {
     USHORT                    usMaxPowerSaveRecoveryTime;
     } WFSCIMPOWERSAVECONTROL, *LPWFSCIMPOWERSAVECONTROL;
```

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

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_POWERSAVETOOSHORT	The power saving mode has not been activated because the device is not able to resume from the power saving mode within the specified *usMaxPowerSaveRecoveryTime* value.
WFS_ERR_CIM_POWERSAVEMEDIAPRESENT	The power saving mode has not been activated because media is present inside the device.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.

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

Value	Meaning
WFS_SRVE_CIM_POWER_SAVE_CHANGE	The power save recovery time has changed.

Comments None.

6.21 WFS_CMD_CIM_REPLENISH

Description This command replenishes items from a single cash unit to multiple cash units. Applications can use this command to ensure that there is the optimum number of items in the cassettes by moving items from a source cash unit to a target cash unit. This is especially applicable if a replenishment cash unit is used for the replenishment and can help to minimize manual replenishment operations.

The WFS_INF_CIM_REPLENISH_TARGET command can be used to determine what cash units can be specified as target cash units for a given source cash unit. Any items which are removed from the source cash unit that are not of the correct currency ID and value for the target cash unit during execution of this command will be returned to the source cash unit.

The *ulCount*, *ulCashInCount*, *ulDispensedCount* and *ulRejectCount* returned with the WFS_INF_CIM_CASH_UNIT_INFO command will be updated as part of the execution of this command. Also for cash recyclers the *ulCount*, *ulDispensedCount* and *ulRejectCount* returned with the WFS_INF_CDM_CASH_UNIT_INFO command will be updated as part of the execution of this command.

If the command fails after some items have been moved, the command will complete with an appropriate error code, and a WFS_EXEE_CIM_INCOMPLETEREPLENISH event will be sent.

Input Param LPWFSCIMREP lpReplenish;

```
typedef struct _wfs_cim_replenish
     {
     USHORT                       usNumberSource;
     LPWFSCIMREPTARGET            *lppReplenishTargets;
     } WFSCIMREP, *LPWFSCIMREP;
```

usNumberSource
Index number of the logical cash unit from which items are to be removed. This is the index number identifier defined in the *usNumber* field of the WFSCIMCASHIN structure of the output data of the WFS_INF_CIM_CASH_UNIT_INFO command.

lppReplenishTargets
Pointer to a NULL-terminated array of pointers to WFSCIMREPTARGET structures. There must be at least one array element:

```
typedef struct_wfs_cim_replenish_target
     {
     USHORT                          usNumberTarget
     ULONG                           ulNumberOfItemsToMove;
     BOOL                            bRemoveAll;
     } WFSCIMREPTARGET, *LPWFSCIMREPTARGET;
```

usNumberTarget
Index number of the logical cash unit to which items are to be moved. This is the index number identifier defined in the *usNumber* field of the WFSCIMCASHIN structure of the output data of the WFS_INF_CIM_CASH_UNIT_INFO command.

ulNumberOfItemsToMove
The number of items to be moved to the target cash unit. Any items which are removed from the source cash unit that are not of the correct currency ID and value for the target cash unit during execution of this command will be returned to the source cash unit. This field will be ignored if the *bRemoveAll* parameter is set to TRUE.

bRemoveAll
Specifies if all items are to be moved to the target cash unit. Any items which are removed from the source cash unit that are not of the correct currency ID and value for the target cash unit during execution of this command will be returned to the source cash unit. If TRUE all items in the source will be moved, regardless of the *ulNumberOfItemsToMove* field value. If FALSE the number of items specified with *ulNumberOfItemsToMove* will be moved.

Output Param LPWFSCIMREPRES lpReplenishResult;

```
typedef struct _wfs_cim_replenish_result
     {
     ULONG                    ulNumberOfItemsRemoved;
     ULONG                    ulNumberOfItemsRejected;
     LPWFSCIMREPTARGETRES     *lppReplenishTargetResults;
     } WFSCIMREPRES, *LPWFSCIMREPRES;
```

ulNumberOfItemsRemoved
Total number of items removed from the source cash unit including rejected items during execution of this command.

ulNumberOfItemsRejected
Total number of items rejected during execution of this command.

lppReplenishTargetResults
Pointer to a NULL-terminated array of pointers to WFSCIMREPTARGETRES structures. In the case where one note type has several releases and these are moved, or where items are moved from a multi denomination cash unit to a multi denomination cash unit, each target can receive several *usNoteID* note types. For example: If one single target was specified with the *lppReplenishTargets* input structure, and this target received two different *usNoteID* note types, then the *lppReplenishTargetResults* array will have two elements. Or if two targets were specified and the first target received two different *usNoteID* note types and the second target received three different *usNoteID* note types, then the *lppReplenishTargetResults* array will have five elements:

```
typedef struct _wfs_cim_replenish_target_result
     {
     USHORT                   usNumberTarget
     USHORT                   usNoteID;
     ULONG                    ulNumberOfItemsReceived;
     } WFSCIMREPTARGETRES, *LPWFSCIMREPTARGETRES;
```

usNumberTarget
Index number of the logical cash unit to which items have been moved. This is the index number identifier defined in the *usNumber* field of the WFSCIMCASHIN structure of the output data of the WFS_INF_CIM_CASH_UNIT_INFO command.

usNoteID
Identification of note type. The note ID represents the note identifiers reported by the WFS_INF_CIM_BANKNOTE_TYPES command.

ulNumberOfItemsReceived
Total number of items received in this target cash unit of the *usNoteID* note type. A zero value will be returned if this target cash unit did not receive any items of this note type, for example due to a cash unit or transport jam.

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

Value	Meaning
WFS_ERR_CIM_CASHUNITERROR	A problem occurred with a cash unit. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details. If appropriate a WFS_EXEE_CIM_INCOMPLETE-REPLENISH event will also be sent.
WFS_ERR_CIM_INVALIDCASHUNIT	The source or target cash unit specified is invalid for this operation. The WFS_INF_CIM_REPLENISH_TARGET command can be used to determine which source or target is valid.
WFS_ERR_CIM_CASHINACTIVE	A cash-in transaction is active.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.

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

Value	Meaning
WFS_USRE_CIM_CASHUNITTHRESHOLD	A threshold condition has occurred in one of the cash units.
WFS_EXEE_CIM_CASHUNITERROR	A problem occurred with a cash unit.
WFS_EXEE_CIM_NOTEERROR	An item detection error has occurred.
WFS_EXEE_CIM_INPUT_P6	Level 2 and / or level 3 notes are detected during this operation.
WFS_EXEE_CIM_INCOMPLETEREPLENISH	If this command fails with an error code (not WFS_SUCCESS) but some items have been moved, then the details will be reported with this event. This event can only occur once per command.

Comments None.

6.22 WFS_CMD_CIM_SET_CASH_IN_LIMIT

Description This command specifies the amount/number of items limitation for the current cash-in transaction. This command can only be called~~ once~~ after the WFS_CMD_CIM_CASH_IN_START command and before the first WFS_CMD_CIM_CASH_IN command, otherwise it will fail with the WFS_ERR_SEQUENCE_ERROR error. Any command that completes the cash-in transaction (i.e. WFS_CMD_CIM_CASH_IN_END, WFS_CMD_CIM_CASH_IN_ROLLBACK, WFS_CMD_CIM_RETRACT and WFS_CMD_CIM_RESET commands) will clear the limit.

This limit is active until the end of the current cash-in transaction. The use of this command is optional, however it needs to be called for each cash-in transaction that needs a limitation.

This command does not disable/enable the recognition of individual note types. The WFS_CMD_CIM_CONFIGURE_NOTETYPES command must be used to refuse a certain note type during cash-in transactions.

If WFS_CIM_LIMITMULTIPLE is specified in the *fwCashInLimit* capability, the command may be called multiple times to add to or override amount limits placed on the current cash-in transaction; the input parameter descriptions below define whether limits are added or overridden. If WFS_CIM_LIMITMULTIPLE is not specified, this command can only be called once per cash-in transaction otherwise it will fail with the WFS_ERR_SEQUENCE_ERROR error.

Input Param LPWFSCIMCASHINLIMIT lpCashInLimit;

Pointer to the WFSCIMCASHINLIMIT structure. This cash-in limit structure can be used to limit the items that can be accepted during the cash-in ~~operation~~transaction. The limit set does not include counterfeit or suspected counterfeit items which may be detected during such a cash-in ~~operation~~transaction. If the *lpCashInLimit* field is set to a NULL pointer there is no specific amount/number of items limit for the ~~next ~~cash-in ~~operation~~transaction and any previously set limits are removed. Note that the cash-in limit set by this command may itself be limited by the physical cash-in limitation of the device.

If one or more limit conditions have been set by this command, the limit reached during the ~~cash-in operation~~WFS_CMD_CIM_CASH_IN command will be reported in the *lpusReason* field of the WFS_EXEE_CIM_INPUTREFUSE event.

```
typedef struct _wfs_cim_cash_in_limit
      {
      ULONG                     ulTotalItemsLimit;
      LPWFSCIMAMOUNTLIMIT       lpAmountLimit;
      } WFSCIMCASHINLIMIT, *LPWFSCIMCASHINLIMIT;
```

ulTotalItemsLimit
If set to a non-zero value, specifies a limit on the total number of items to be accepted during the cash-in ~~operation~~transaction. If set to a zero value, this limitation will not be performed.

This limitation can only be used if WFS_CIM_LIMITBYTOTALITEMS is specified in the *fwCashInLimit* field of the WFS_INF_CIM_CAPABILITIES command. If ~~however this~~*ulTotalItemsLimit* is ~~specified~~non-zero but not supported the WFS_ERR_UNSUPP_DATA error will be returned and no limit will be set.

This parameter overrides any previously set limit on the total number of items.

lpAmountLimit
Pointer to the WFSCIMAMOUNTLIMIT structure.~~ If set to a NULL pointer this limitation will not be performed. For CIM devices which can accept more than one currency this limit can only be applied to one currency for each cash-in operation.~~

This limitation can only be used if WFS_CIM_LIMITBYAMOUNT is ~~specified~~reported in the *fwCashInLimit* field of the WFS_INF_CIM_CAPABILITIES command. If ~~however this~~*lpAmountLimit* is ~~specified~~not NULL but not supported the WFS_ERR_UNSUPP_DATA error will be returned and no limit will be set.

If *lpAmountLimit* is set to a NULL pointer, this has no impact.

If *lpAmountLimit* is not NULL, this specifies the maximum amount of the currency specified by *cCurrencyID* which can be accepted in the current cash-in transaction. If the currency has already been specified for the current cash-in transaction, the maximum amount is overridden for that currency. If the currency has not already been specified, it is added to a set of currency specific limits to apply to the cash-in transaction. If any currency limits are specified for the current cash-in transaction, the handling of other currencies is dependent on whether the WFS_CIM_LIMITREFUSEOTHER flag is reported in the *fwCashInLimit* field of the WFS_INF_CIM_CAPABILITIES command. See Comments below for examples.

```
typedef struct _wfs_cim_amount_limit
      {
      CHAR                       cCurrencyID[3];
      ULONG                      ulAmount;
      } WFSCIMAMOUNTLIMIT, *LPWFSCIMAMOUNTLIMIT;
```

cCurrencyID
Currency identifier in ISO 4217 format [Ref. 2]. This must not be three ASCII 0x20 characters.

ulAmount
If set to a non-zero value, specifies a limit on the total amount of the cash-in ~~operation~~transaction for the specified *cCurrencyID*. This value is expressed in minimum dispense units (see section WFS_INF_CIM_CURRENCY_EXP). If set to a zero value, ~~this limitation~~no amount limit will ~~not be performed~~apply to the specified currency.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.

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

Comments Where a CIM device can accept multiple currencies, this command can be called several times to specify the limits for each individual currency if WFS_CIM_LIMITMULTIPLE is reported in the *fwCashInLimit* capability. The following examples illustrate different limits set on cash-in transactions on a CIM device which can accept EUR, GBP and USD and shows that both amount and total number of items limits can be specified for a single transaction.

If the WFS_CIM_LIMITREFUSEOTHER flag is reported in the *fwCashInLimit* field of the WFS_INF_CIM_CAPABILITIES command, if any currency amounts are specified, any currencies not specified are refused. If the WFS_CIM_LIMITREFUSEOTHER flag is not reported, then unspecified currencies are accepted, therefore a currency may only be completely refused if all its note types are disabled using WFS_CMD_CIM_CONFIGURE_NOTETYPES.

Transaction limits	WFS_CMD_CIM_SET_CASH_IN_LIMIT calls (*ulTotalItemsLimit, cCurrencyID, ulAmount*)
EUR 100 or GBP 200 or USD 500 Maximum number of items allowed limited by physical capability	*0, EUR, 100* *0, GBP, 200* *0, USD, 500*
EUR 100 or GBP 200, USD handled per WFS_CIM_LIMITREFUSEOTHER definition Maximum 50 items allowed	*50, EUR, 100* *50, GBP, 200*
USD 500, other currencies handled per WFS_CIM_LIMITREFUSEOTHER definition Maximum number of items allowed limited by physical capability	*0, USD, 500*

EUR limited by physical capability of the device. Other currencies handled per WFS_CIM_LIMITREFUSEOTHER definition	*0, EUR, 0*
EUR limited by physical capability of the device GBP 100, USD handled per WFS_CIM_LIMITREFUSEOTHER definition	*0, EUR, 0* *0, GBP, 100*

~~**Comments**~~ ~~None.~~

6.23 WFS_CMD_CIM_CASH_UNIT_COUNT

Description This command counts the items in the cash unit(s). If it is necessary to move items internally to count them, the items should be returned to the cash unit from which they originated before completion of the command. If items could not be moved back to the cash unit they originated from and did not get rejected, the command will complete with an appropriate error.

During the execution of this command one WFS_SRVE_CIM_CASHUNITINFOCHANGED event will be generated for each cash unit that has been counted successfully, or if the counts have changed, even if the overall command fails.

After completion of this command the number of items rejected can be determined by calling the WFS_INF_CIM_CASH_UNIT_INFO command and checking the value of the *ulRejectCount* field within the WFSCIMCASHIN structure and WFSCIMPHCU substructures. The *ulRejectCount* value is incremented by one for each item rejected during execution of this command.

This command is designed to be used on CIM devices where the *ulCount* cannot be guaranteed to be accurate and therefore may need to be automatically counted periodically. Upon successful completion, for those cash units that have been counted, the *ulCount* field within the WFSCIMCASHIN structure and its WFSCIMNOTENUMBERLIST and WFSCIMPHCU substructures are accurately reported with the WFS_INF_CIM_CASH_UNIT_INFO command.

Input Param LPWFSCIMCOUNT lpCount;

If the *fwCountActions* WFS_CIM_COUNTINDIVIDUAL capability is supported, this structure can provide data indicating which cash units are to be counted. If the *fwCountActions* WFS_CIM_COUNTALL capability is supported, this pointer can be NULL, and all cash units will be counted.

```
typedef struct _wfs_cim_count
     {
     USHORT                    usCount;
     LPUSHORT                  lpusCUNumList;
     } WFSCIMCOUNT, *LPWFSCIMCOUNT;
```

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

lpusCUNumList
Pointer to an array of USHORT values containing the logical numbers of the individual cash units to be counted. All physical cash units which the logical cash unit is composed of will be counted. If an invalid logical number is contained in this list, the command will fail with a WFS_ERR_CIM_CASHUNITERROR error.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_INVALIDCASHUNIT	At least one of the logical cash units specified is either invalid or does not support being counted. No cash units have been counted.
WFS_ERR_CIM_CASHINACTIVE	A cash-in transaction is active.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.
WFS_ERR_CIM_TOOMANYITEMSTOCOUNT	There were too many items. The required internal position may have been of insufficient size. All items should be returned to the cash unit from which they originated.
WFS_ERR_CIM_COUNTPOSNOTEMPTY	A required internal position is not empty so a cash unit count is not possible.

WFS_ERR_CIM_CASHUNITERROR	A cash unit caused a problem. A WFS_EXEE_CIM_CASHUNITERROR event will be posted with the details.

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

Value	Meaning
WFS_SRVE_CIM_CASHUNITINFOCHANGED	The counting of a cash unit has completed or the counts have changed.
WFS_SRVE_CIM_CASHUNITTHRESHOLD	A threshold condition has occurred in one of the cash units.
WFS_EXEE_CIM_CASHUNITERROR	A problem occurred with a cash unit.
WFS_EXEE_CIM_NOTEERROR	An item detection error has occurred.
WFS_EXEE_CIM_INPUT_P6	Level 2 and / or level 3 notes are detected during this operation.

Comments None.

6.24 WFS_CMD_CIM_DEVICE_LOCK_CONTROL

Description This command can be used to lock or unlock a CIM device, it can also be used to lock or unlock one or more cash units.

During normal device operation the device and cash units will be locked and removal will not be possible. If supported the device or cash units can be unlocked, ready for removal. In this situation the device will still remain online and cash-in or dispense operations will be possible, as long as the device or cash units are not physically removed from their normal operating position.

If the lock action is specified and the device or cash units are already locked, or if the unlock action is specified and the device or cash units are already unlocked then the action will complete successfully.

Once a cash unit has been removed and reinserted it will then have a WFS_CIM_STATCUMANIP status. This status can only be cleared by issuing a WFS_CMD_CIM_START_EXCHANGE/WFS_CMD_CIM_END_EXCHANGE command sequence.

The device and all cash units will also be locked implicitly as part of the execution of the WFS_CMD_CIM_END_EXCHANGE or the WFS_CMD_CIM_RESET command.

Input Param LPWFSCIMDEVICELOCKCONTROL lpDeviceLockControl;

```
typedef struct _wfs_cim_device_lock_control
     {
     WORD                         wDeviceAction;
     WORD                         wCashUnitAction;
     LPWFSCIMUNITLOCKCONTROL      *lppUnitLockControl;
     } WFSCIMDEVICELOCKCONTROL, *LPWFSCIMDEVICELOCKCONTROL;
```

wDeviceAction
Specifies to lock or unlock the CIM device in its normal operating position. Possible values are:

Value	Meaning
WFS_CIM_LOCK	Locks the CIM device so that it cannot be removed from its normal operating position.
WFS_CIM_UNLOCK	Unlocks the CIM device so that it can be removed from its normal operating position.
WFS_CIM_NOLOCKACTION	No lock/unlock action will be performed on the CIM device.

wCashUnitAction
Specifies the type of lock/unlock action on physical cash units as one of the following values:

Value	Meaning
WFS_CIM_LOCKALL	Locks all physical cash units supported.
WFS_CIM_UNLOCKALL	Unlocks all physical cash units supported.
WFS_CIM_LOCKINDIVIDUAL	Locks/unlocks physical cash units individually as specified in the *lppUnitLockControl* parameter.
WFS_CIM_NOLOCKACTION	No lock/unlock action will be performed on cash units.

lppUnitLockControl
Pointer to a NULL-terminated array of pointers to WFSCIMUNITLOCKCONTROL structures; only valid in the case where WFS_CIM_LOCKINDIVIDUAL is specified in the *wCashUnitAction* field. Otherwise this field will be ignored. Each element specifies one cash unit to be locked/unlocked:

```
typedef struct _wfs_cim_unit_lock_control
     {
     LPSTR                        lpPhysicalPositionName;
     WORD                         wUnitAction;
     } WFSCIMUNITLOCKCONTROL, *LPWFSCIMUNITLOCKCONTROL;
```

lpPhysicalPositionName
Specifies which physical cash unit is to be locked/unlocked. This name is the same as the *lpPhysicalPositionName* in the WFSCIMPHCU structure. Only physical cash units reported by the WFS_INF_CIM_DEVICELOCK_STATUS command can be specified.

wUnitAction
Specifies whether to lock or unlock the physical cash unit indicated in the *lpPhysicalPositionName* parameter. Possible values are:

Value	Meaning
WFS_CIM_LOCK	Locks the specified cash unit so that it cannot be removed from the CIM device.
WFS_CIM_UNLOCK	Unlocks the specified cash unit so that it can be removed from the CIM device.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_INVALIDCASHUNIT	The cash unit type specified is invalid.
WFS_ERR_CIM_CASHINACTIVE	A cash-in transaction is active.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM service is in an exchange state.
WFS_ERR_CIM_DEVICELOCKFAILURE	The device and/or the cash units specified could not be locked/unlocked. (e.g. the lock action could not be performed because the cash unit specified to be locked had been removed).

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

Value	Meaning
WFS_USRE_CIM_CASHUNITTHRESHOLD	A threshold condition has occurred in one of the cash units.
WFS_EXEE_CIM_CASHUNITERROR	A problem occurred with a cash unit.

Comments The normal command sequence is as follows:

Step1: WFS_CMD_CIM_DEVICE_LOCK_CONTROL command is executed to unlock the device and some or all of the cash units.

Step 2: Optionally a WFS_CMD_CIM_CASH_IN_START / WFS_CMD_CIM_CASH_IN / WFS_CMD_CIM_CASH_IN_END cash-in transaction or a WFS_CMD_CDM_DISPENSE / WFS_CMD_CDM_PRESENT transaction on a cash recycler device may be performed.

Step 3: The operator was not required to remove any of the cash units, all cash units are still in their original position.

Step 4: WFS_CMD_CIM_DEVICE_LOCK_CONTROL command is executed to lock the device and the cash units.

The relation of lock/unlock control with the WFS_CMD_CIM_START_EXCHANGE and the WFS_CMD_CIM_END_EXCHANGE commands is as follows:

Step 1: WFS_CMD_CIM_DEVICE_LOCK_CONTROL command is executed to unlock the device and some or all of the cash units.

Step 2: Optionally a WFS_CMD_CIM_CASH_IN_START / WFS_CMD_CIM_CASH_IN / WFS_CMD_CIM_CASH_IN_END cash-in transaction or a WFS_CMD_CDM_DISPENSE / WFS_CMD_CDM_PRESENT transaction on a cash recycler device may be performed.

Step 3: The operator removes and reinserts one or multiple of the previously unlocked cash units. The associated WFS_SRVE_CIM_CASHUNITINFOCHANGED event will be posted and after the reinsertion the cash unit will show the status WFS_CIM_STATCUMANIP.

Step 4: WFS_CMD_CIM_START_EXCHANGE command is executed.

Step 5: WFS_CMD_CIM_END_EXCHANGE command is executed. During this command execution the Service Provider implicitly locks the device and all previously unlocked cash units. The cash unit status of the previously removed cash unit will be reset.

6.25 WFS_CMD_CIM_SET_MODE

Description This execute command is used to set the deposit mode for the device and is only applicable for Mixed Media processing. The deposit mode determines how the device will process non cash items that are inserted. The deposit mode applies to all subsequent transactions. The deposit mode is persistent and is unaffected by a device reset by WFS_CMD_CIM_RESET or reset on another interface. The command will fail with a WFS_ERR_INVALID_DATA error where an attempt is made to set a mode that is not supported.

Input Param LPWFSCIMSETMODE lpMode;

```
typedef struct _wfs_cim_setmode
     {
     WORD                          wMixedMode;
     } WFSCIMSETMODE, *LPWFSCIMSETMODE;
```

wMixedMode
Specifies the Mixed Media mode of the device as one of the following values:

Value	Meaning
WFS_CIM_MIXEDMEDIANOTACTIVE	Mixed Media transactions are deactivated. This is the default mode.
WFS_CIM_IPMMIXEDMEDIA	Mixed Media transactions are activated in combination with the IPM interface as defined by the capability *wMixedMode*.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_CASHINACTIVE	A cash-in transaction is active.
WFS_ERR_CIM_MEDIAINACTIVE	An item processing transaction is active.

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

Comments The commands WFS_CMD_CIM_SET_MODE and WFS_CMD_IPM_SET_MODE are equivalent; an application can use either to control the Mixed Media mode. If the requested mode is already active WFS_CMD_CIM_SET_MODE command returns with WFS_SUCCESS.

6.26 WFS_CMD_CIM_PRESENT_MEDIA

Description This command opens the shutter and presents items to be taken by the customer. The shutter is automatically closed after the media is taken. The command can be called after a WFS_CMD_CIM_CASH_IN, WFS_CMD_CIM_ROLLBACK, WFS_CMD_CIM_RESET or WFS_CMD_CIM_CREATE_P6_SIGNATURE command and can be used with explicit and implicit shutter control. The command is only valid on positions where *fwUsage* reported by the WFS_INF_CIM_POSITION_CAPABILITIES command is WFS_CIM_POSROLLBACK or WFS_CIM_POSREFUSE and where *bPresentControl* reported by the WFS_INF_CIM_POSITION_CAPABILITIES command is FALSE.

This command cannot be used to present items stacked through the CDM interface. Where this is attempted the command fails with a WFS_ERR_SEQUENCE_ERROR error.

Mixed Media Mode:

If the device is operating in Mixed Media mode (WFSCIMSTATUS.*wMixedMode* == WFS_CIM_IPMMIXEDMEDIA) this command will not perform any operation unless the WFS_CMD_IPM_PRESENT_MEDIA command is called or has already been called on the IPM interface. Shutter control on devices that support Mixed Media processing is always implicit.

Input Param LPWFSCIMPRESENT lpPresent;

If the input parameter is NULL then all refused items are returned from all positions in a sequence determined by the Service Provider.

```
typedef struct _wfs_cim_present
      {
      WORD                         fwPosition;
      } WFSCIMPRESENT, *LPWFSCIMPRESENT;
```

fwPosition
Describes the position where the media is to be presented as one of the following values:

Value	Meaning
WFS_CIM_POSNULL	The default configuration information should be used.
WFS_CIM_POSINLEFT	Present items to the left input position.
WFS_CIM_POSINRIGHT	Present items to the right input position.
WFS_CIM_POSINCENTER	Present items to of the center input position.
WFS_CIM_POSINTOP	Present items to the top input position.
WFS_CIM_POSINBOTTOM	Present items to the bottom input position.
WFS_CIM_POSINFRONT	Present items to the front input position.
WFS_CIM_POSINREAR	Present items to the rear input position.
WFS_CIM_POSOUTLEFT	Present items to the left output position.
WFS_CIM_POSOUTRIGHT	Present items to the right output position.
WFS_CIM_POSOUTCENTER	Present items to the center output position.
WFS_CIM_POSOUTTOP	Present items to the top output position.
WFS_CIM_POSOUTBOTTOM	Present items to the bottom output position.
WFS_CIM_POSOUTFRONT	Present items to the front output position.
WFS_CIM_POSOUTREAR	Present items to of the rear output position.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_UNSUPPOSITION	The position specified is not supported or is not a valid position for this command.
WFS_ERR_CIM_SHUTTERNOTOPEN	Shutter failed to open.
WFS_ERR_CIM_NOITEMS	There were no items to present at the specified position.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.
WFS_ERR_CIM_FOREIGN_ITEMS_DETECTED	Foreign items have been detected in the input position.

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

Value	Meaning
WFS_SRVE_CIM_ITEMSTAKEN	The items have been removed by the user. This event is only generated if the *bItemsTakenSensor* field returned in the capabilities information is TRUE.
WFS_SRVE_CIM_ITEMSPRESENTED	Items have been presented to the user to be taken.
WFS_SRVE_CIM_SHUTTERSTATUSCHANGED	The shutter status has changed.

Comments None.

6.27 WFS_CMD_CIM_DEPLETE

Description This command removes items from multiple cash units to a single cash unit. Applications can use this command to ensure that there is the optimum number of items in the cassettes by moving items from source cash units to a target cash unit. This is especially applicable if surplus items are removed from multiple recycle cash units to a replenishment cash unit and can help to minimize manual replenishment operations.

The WFS_INF_CIM_DEPLETE_SOURCE command can be used to determine what cash units can be specified as source cash units for a given target cash unit.

The *ulCount*, *ulCashInCount*, *ulDispensedCount* and *ulRejectCount* returned with the WFS_INF_CIM_CASH_UNIT_INFO command will be updated as part of the execution of this command. Also for cash recyclers the *ulCount*, *ulDispensedCount* and *ulRejectCount* returned with the WFS_INF_CDM_CASH_UNIT_INFO command will be updated as part of the execution of this command.

If the command fails after some items have been moved, the command will complete with an appropriate error code, and a WFS_EXEE_CIM_INCOMPLETEDEPLETE event will be sent.

Input Param LPWFSCIMDEP lpDeplete;

```
typedef struct _wfs_cim_deplete
      {
      LPWFSCIMDEPSOURCE          *lppDepleteSources;
      USHORT                     usNumberTarget;
      } WFSCIMDEP, *LPWFSCIMDEP;
```

lppDepleteSources
Pointer to a NULL-terminated array of pointers to WFSCIMDEPSOURCE structures. There must be at least one WFSCIMDEPSOURCE structure:

```
typedef struct_wfs_cim_deplete_source
      {
      USHORT                          usNumberSource;
      ULONG                           ulNumberOfItemsToMove;
      BOOL                            bRemoveAll;
      } WFSCIMDEPSOURCE, *LPWFSCIMDEPSOURCE;
```

usNumberSource
Index number of the logical cash unit from which items are to be removed. This is the index number identifier defined in the *usNumber* field of the WFSCIMCASHIN structure of the output data of the WFS_INF_CIM_CASH_UNIT_INFO command.

ulNumberOfItemsToMove
The number of items to be moved from the source cash unit. This must be equal to or less than the count of items reported for the cash unit specified by *usNumberSource*. This field will be ignored if the *bRemoveAll* parameter is set to TRUE.

bRemoveAll
Specifies if all items are to be moved from the source cash unit. If TRUE all items in the source will be moved, regardless of the *ulNumberOfItemsToMove* field value. If FALSE the number of items specified with *ulNumberOfItemsToMove* will be moved.

usNumberTarget
Index number of the logical cash unit to which items are to be moved. This is the index number identifier defined in the *usNumber* field of the WFSCIMCASHIN structure of the output data of the WFS_INF_CIM_CASH_UNIT_INFO command.

Output Param LPWFSCIMDEPRES lpDepleteResult;

```
typedef struct _wfs_cim_deplete_result
      {
      ULONG                      ulNumberOfItemsReceived;
      ULONG                      ulNumberOfItemsRejected;
      LPWFSCIMDEPSOURCERES       *lppDepleteSourceResults;
      } WFSCIMDEPRES, *LPWFSCIMDEPRES;
```

ulNumberOfItemsReceived
Total number of items received in the target cash unit during execution of this command.

ulNumberOfItemsRejected
Total number of items rejected during execution of this command.

lppDepleteSourceResults
Pointer to a NULL-terminated array of pointers to WFSCIMDEPSOURCERES structures. In the case where one item type has several releases and these are moved, or where items are moved from a multi denomination cash unit to a multi denomination cash unit, each source can move several *usNoteID* item types. For example: If one single source was specified with the *lppDepleteSources* input structure, and this source moved two different *usNoteID* item types, then the *lppDepleteSourceResults* array will have two elements. Or if two sources were specified and the first source moved two different *usNoteID* item types and the second source moved three different *usNoteID* item types, then the *lppDepleteSourceResults* array will have five elements:

```
typedef struct _wfs_cim_deplete_source_result
     {
     USHORT                     usNumberSource;
     USHORT                     usNoteID;
     ULONG                      ulNumberOfItemsRemoved;
     } WFSCIMDEPSOURCERES, *LPWFSCIMDEPSOURCERES;
```

usNumberSource
Index number of the logical cash unit from which items have been removed. This is the index number identifier defined in the *usNumber* field of the WFSCIMCASHIN structure of the output data of the WFS_INF_CIM_CASH_UNIT_INFO command.

usNoteID
Identification of item type. The note ID represents the item identifiers reported by the WFS_INF_CIM_BANKNOTE_TYPES command.

ulNumberOfItemsRemoved
Total number of items removed from this source cash unit of the *usNoteID* item type. A zero value will be returned if this source cash unit did not move any items of this item type, for example due to a cash unit or transport jam.

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

Value	Meaning
WFS_ERR_CIM_CASHUNITERROR	A problem occurred with a cash unit. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details. If appropriate a WFS_EXEE_CIM_INCOMPLETE-DEPLETE event will also be sent.
WFS_ERR_CIM_INVALIDCASHUNIT	The source or target cash unit specified is invalid for this operation. The WFS_INF_CIM_DEPLETE_SOURCE command can be used to determine which source or target is valid.
WFS_ERR_CIM_CASHINACTIVE	A cash-in transaction is active.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.

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

Value	Meaning
WFS_USRE_CIM_CASHUNITTHRESHOLD	A threshold condition has occurred in one of the cash units.
WFS_EXEE_CIM_CASHUNITERROR	A problem occurred with a cash unit.
WFS_EXEE_CIM_NOTEERROR	An item detection error has occurred.
WFS_EXEE_CIM_INPUT_P6	Level 2 and / or level 3 notes are detected during this operation.
WFS_EXEE_CIM_INCOMPLETEDEPLETE	If this command fails with an error code (not WFS_SUCCESS) but some items have been moved, then the details will be reported with this event. This event can only occur once per command.

Comments	None.

6.28 WFS_CMD_CIM_SET_BLACKLIST

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

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

LPWFSCIMBLACKLIST lpBlacklist;

The LPWFSCIMBLACKLIST structure is defined in the documentation of the WFS_INF_CIM_GET_BLACKLIST command.

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

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

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

The WFSCIMBLACKLISTELEMENT structure is defined in the documentation of the WFS_INF_CIM_GET_BLACKLIST command.

> *lpszSerialNumber*
> This Unicode string defines the serial number or a mask of serial numbers of one blacklist element with the defined currency and value. For a definition of the mask see section ~~2~~4.
>
> *cCurrencyID*
> The three character ISO format currency identifier [Ref. 2] of the blacklist element.
>
> *ulValue*
> The value of a blacklist element. This field can be set to zero to match all values.

Output Param None.

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

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

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

6.29 WFS_CMD_CIM_SYNCHRONIZE_COMMAND

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

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

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

Input Param LPWFSCIMSYNCHRONIZECOMMAND lpSynchronizeCommand;

```
typedef struct _wfs_cim_synchronize_command
     {
     DWORD                    dwCommand;
     LPVOID                   lpCmdData;
     } WFSCIMSYNCHRONIZECOMMAND, *LPWFSCIMSYNCHRONIZECOMMAND;
```

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

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

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

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM is in an exchange state.
WFS_ERR_CIM_COMMANDUNSUPP	The command specified in the *dwCommand* field is not supported by the Service Provider.
WFS_ERR_CIM_SYNCHRONIZEUNSUPP	The preparation for the command specified in the *dwCommand* with the parameter specified in the *lpCmdData* is not supported by the Service Provider.

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

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

6.30 WFS_CMD_CIM_SET_CLASSIFICATION_LIST

Description	This command is used to specify the entire note classification list. Any items not specified in this list will be handled according to normal classification rules. This information is persistent. Information set by this command overrides any existing blacklist or classification list, although it is not recommended that an application use both this command and WFS_CMD_CIM_SET_BLACKLIST to avoid overlap and confusion. If a note is reclassified, it is handled as though it was a note of the new classification. For example, a fit note reclassified as unfit would be treated as though it were unfit, which may mean that the note is not dispensed. Reclassification cannot be used to change a note's classification to a higher level, for example, a note recognized as counterfeit by the device cannot be reclassified as genuine. In addition, it is not possible to re-classify a level 2 note as level 1. If two or more classification elements specify overlapping note definitions, but different *usLevel* values then the first one takes priority.
Input Param	LPWFSCIMCLASSIFICATIONLIST lpClassificationList; The LPWFSCIMCLASSIFICATIONLIST structure is defined in WFS_INF_CIM_GET_CLASSIFICATION_LIST. This parameter should be set to NULL if the application wishes to empty the note classification list.
Output Param	None.
Error Codes	Only the generic error codes defined in [Ref. 1] can be generated by this command.
Events	Only the generic events defined in [Ref. 1] can be generated by this command.
Comments	None.

6.31 WFS_CMD_CIM_PREPARE_PRESENT

Description In cases where multiple bunches are to be returned under explicit shutter control, this command is used for the purpose of moving a remaining bunch to the output position explicitly before using the following commands:

WFS_CMD_CIM_OPEN_SHUTTER
WFS_CMD_CIM_PRESENT_MEDIA

The application can tell whether the additional items were left by using WFS_INF_CIM_PRESENT_STATUS command.

This command does not affect the status of the current cash-in transaction.

Input Param LPWFSCIMMOVEITEMS lpPresent;

```
typedef struct _wfs_cim_moveitems
        {
        WORD                              fwPosition;
        } WFSCIMMOVEITEMS, *LPWFSCIMMOVEITEMS;
```

fwPosition
Describes the position where the items are to be moved as one of the following values:

Value	Meaning
WFS_CIM_POSNULL	The default configuration information should be used.
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.

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_UNSUPPOSITION	The position specified is not supported or is not a valid position for this command.
WFS_ERR_CIM_POSITION_NOT_EMPTY	The input or output position is not empty.
WFS_ERR_CIM_NOITEMS	There were no items to present at the specified position.
WFS_ERR_CIM_CASHUNITERROR	A cash unit caused a problem. A WFS_EXEE_CIM_CASHUNITERROR event will be posted with the details.

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

Value	Meaning
WFS_EXEE_CIM_CASHUNITERROR	A problem occurred with the cash unit.
WFS_EXEE_CIM_INPUT_P6	Level 2 and / or level 3 notes are detected.
WFS_EXEE_CIM_INFO_AVAILABLE	Information is available for items detected during the cash processing operation.

Comments None.

7. Events

7.1 WFS_SRVE_CIM_SAFEDOOROPEN

Description	This service event specifies that the safe door has been opened.
Event Param	None.
Comments	None.

7.2 WFS_SRVE_CIM_SAFEDOORCLOSED

Description	This service event specifies that the safe door has been closed.
Event Param	None.
Comments	None.

7.3 WFS_USRE_CIM_CASHUNITTHRESHOLD

Description This user event is generated when a threshold condition has occurred in one of the logical cash units or the threshold condition is removed. If the logical cash unit is a shared cash unit in a compound device then this event can also be generated as a result of an operation on another device class.

This event can be triggered either by hardware sensors in the device or by the logical *ulCount* reaching the *ulMaximum* value as specified in the WFSCIMCASHIN structure. For a cash unit of type WFS_CIM_TYPERETRACTCASSETTE, it is also possible that this event can instead be triggered by the *ulCashInCount* reaching the *ulMaximum* value. For more detail see the *bRetractNoteCountThresholds* field description in the WFS_INF_CIM_CASH_UNIT_CAPABILITIES command.

The application can check if the device has hardware sensors by querying the *bHardwareSensors* field of the WFSCIMPHCUCAPABILITIES structure. If any of the physical cash units associated with the logical cash unit have this capability then threshold events based on hardware sensors will be triggered if the *ulMaximum* values are not used and are set to zero.

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

Event Param LPWFSCIMCASHIN lpCashUnit;

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

Comments None.

7.4 WFS_SRVE_CIM_CASHUNITINFOCHANGED

Description This service event is generated under the following circumstances:

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

  WFS_CMD_CIM_SET_CASH_UNIT_INFO
  WFS_CMD_CIM_END_EXCHANGE
- In addition this event will be generated when a cash unit has been counted during the WFS_CMD_CIM_CASH_UNIT_COUNT command execution.

If the cash unit is a shared cash unit in a compound device then this event can also be generated as a result of an operation on another device class.

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

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

If a physical cash unit of a new logical cash unit inserted the cash unit structure reported by the last WFS_INF_CIM_CASH_UNIT_INFO command is no longer valid. In that case an application should issue a WFS_INF_CIM_CASH_UNIT_INFO command after receiving this event to obtain updated cash unit information.

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.

7.5 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.

7.6 WFS_EXEE_CIM_CASHUNITERROR

Description This execute event specifies that a cash unit was addressed which caused a problem.

Event Param 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:

Value	Meaning
WFS_CIM_CASHUNITEMPTY	Specified cash unit is empty.
WFS_CIM_CASHUNITERROR	Specified cash unit has malfunctioned.
WFS_CIM_CASHUNITFULL	Specified cash unit is full.
WFS_CIM_CASHUNITLOCKED	The *bAppLock* field of the WFSCIMCASHIN structure has previously been set to TRUE and the cash unit remains locked.
WFS_CIM_CASHUNITNOTCONF	Specified cash unit is not configured due to being removed and/or replaced with a different cash unit.
WFS_CIM_CASHUNITINVALID	Specified cash unit is invalid.
WFS_CIM_CASHUNITCONFIG	Attempt to change the setting of a self-configuring cash unit.
WFS_CIM_FEEDMODULEPROBLEM	A problem has been detected with the feeding module.
WFS_CIM_CASHUNITPHYSICALLOCKED	The cash unit could not be unlocked by the WFS_CMD_CIM_DEVICE_LOCK_-CONTROL command and remains physically locked.
WFS_CIM_CASHUNITPHYSICALUNLOCKED	The cash unit could not be locked by the WFS_CMD_CIM_DEVICE_LOCK_-CONTROL command and remains physically unlocked.

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.

7.7 WFS_SRVE_CIM_ITEMSTAKEN

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

Event Param LPWFSCIMPOSITIONINFO lpPositionInfo;

```
typedef struct _wfs_cim_position_info
     {
     WORD                          wPosition;
     WORD                          wAdditionalBunches;
     USHORT                        usBunchesRemaining;
     } WFSCIMPOSITIONINFO, *LPWFSCIMPOSITIONINFO;
```

wPosition
Specifies the position from which the items have been taken, set to one of the following values:

Value	Meaning
WFS_CIM_POSINLEFT	Items taken from the left input position.
WFS_CIM_POSINRIGHT	Items taken from the right input position.
WFS_CIM_POSINCENTER	Items taken from the center input position.
WFS_CIM_POSINTOP	Items taken from the top input position.
WFS_CIM_POSINBOTTOM	Items taken from the bottom input position.
WFS_CIM_POSINFRONT	Items taken from the front input position.
WFS_CIM_POSINREAR	Items taken from the rear input position.
WFS_CIM_POSOUTLEFT	Items taken from the left output position.
WFS_CIM_POSOUTRIGHT	Items taken from the right output position.
WFS_CIM_POSOUTCENTER	Items taken from the center output position.
WFS_CIM_POSOUTTOP	Items taken from the top output position.
WFS_CIM_POSOUTBOTTOM	Items taken from the bottom output position.
WFS_CIM_POSOUTFRONT	Items taken from the front output position.
WFS_CIM_POSOUTREAR	Items taken from the rear output position.

wAdditionalBunches
This value will always be zero within this event.

usBunchesRemaining
This value will always be zero within this event.

Comments None.

7.8 WFS_SRVE_CIM_COUNTS_CHANGED

Description This service event is generated if the device is a compound device and the counts in a shared cash unit have changed as a result of an operation on the other device class other than as a result of an operation that explicitly sets counts. For example, WFS_CMD_CDM_SET_CASH_UNIT_INFO and WFS_CMD_CDM_END_EXCHANGE commands on the CDM and WFS_CMD_IPM_SET_MEDIA_BIN_INFO command on the IPM.

Event Param LPWFSCIMCOUNTSCHANGED lpCountsChanged;

```
typedef struct _wfs_cim_counts_changed
     {
     USHORT                    usCount;
     LPUSHORT                  lpusCUNumList;
     } WFSCIMCOUNTSCHANGED, *LPWFSCIMCOUNTSCHANGED;
```

usCount
The size of *lpusCUNumList*.

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

Comments None.

7.9 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 LPUSHORT lpusReason;

lpusReason
Pointer to an USHORT holding the reason for refusing a part of the amount. Possible values are:

Value	Meaning
WFS_CIM_CASHINUNITFULL	Cash unit is full.
WFS_CIM_INVALIDBILL	Recognition of the items took place, but one or more of the items are invalid.
WFS_CIM_NOBILLSTODEPOSIT	There are no items in the input area.
WFS_CIM_DEPOSITFAILURE	A deposit has failed for a reason not covered by the other reasons and the failure is not a fatal hardware problem, for example failing to pick an item from the input area.
WFS_CIM_COMMINPCOMPFAILURE	Failure of a common input component which is shared by all cash units.
WFS_CIM_STACKERFULL	The intermediate stacker is full.
WFS_CIM_FOREIGN_ITEMS_DETECTED	Foreign items have been detected in the input position.
WFS_CIM_INVALIDBUNCH	Recognition of the items did not take place. The bunch of notes inserted is invalid, e.g. it is too large or was inserted incorrectly.
WFS_CIM_COUNTERFEIT	One or more counterfeit items have been detected and refused. This is only applicable ~~to devices which do~~where notes are not ~~support a legislative note handling standard~~classified as level 2 and ~~are~~the device is capable of differentiating between invalid and counterfeit items.
WFS_CIM_LIMITOVERTOTALITEMS	Number of items count exceeded the limitation set with the WFS_CMD_CIM_SET_CASH_IN_LIMIT command.
WFS_CIM_LIMITOVERAMOUNT	Amount exceeded the limitation set with the WFS_CMD_CIM_SET_CASH_IN_LIMIT command.

Comments None.

7.10 WFS_SRVE_CIM_ITEMSPRESENTED

Description This service event specifies that items have been presented to the output position, and the shutter has been opened to allow the user to take the items.

Event Param LPWFSCIMPOSITIONINFO lpPositionInfo;

```
typedef struct _wfs_cim_position_info
     {
     WORD                          wPosition;
     WORD                          wAdditionalBunches;
     USHORT                        usBunchesRemaining;
     } WFSCIMPOSITIONINFO, *LPWFSCIMPOSITIONINFO;
```

wPosition
Specifies the position from which the items have been presented, set to one of the following values:

Value	Meaning
WFS_CIM_POSOUTLEFT	Items presented at the left output position.
WFS_CIM_POSOUTRIGHT	Items presented at the right output position.
WFS_CIM_POSOUTCENTER	Items presented at the center output position.
WFS_CIM_POSOUTTOP	Items presented at the top output position.
WFS_CIM_POSOUTBOTTOM	Items presented at the bottom output position.
WFS_CIM_POSOUTFRONT	Items presented at the front output position.
WFS_CIM_POSOUTREAR	Items presented at the rear output position.
WFS_CIM_POSINLEFT	Items presented at the left input position.
WFS_CIM_POSINRIGHT	Items presented at the right input position.
WFS_CIM_POSINCENTER	Items presented at the center input position.
WFS_CIM_POSINTOP	Items presented at the top input position.
WFS_CIM_POSINBOTTOM	Items presented at the bottom input position.
WFS_CIM_POSINFRONT	Items presented at the front input position.
WFS_CIM_POSINREAR	Items presented at the rear input position.

wAdditionalBunches
Specifies whether or not additional bunches of items are remaining to be presented as a result of the current operation, set to one of the following values:

Value	Meaning
WFS_CIM_ADDBUNCHNONE	No additional bunches remain.
WFS_CIM_ADDBUNCHONEMORE	At least one additional bunch remains.
WFS_CIM_ADDBUNCHUNKNOWN	It is unknown whether additional bunches remain.

usBunchesRemaining
If *wAdditionalBunches* is WFS_CIM_ADDBUNCHONEMORE, specifies the number of additional bunches of items remaining to be presented as a result of the current operation. If the number of additional bunches is at least one, but the precise number is unknown, *usBunchesRemaining* will be WFS_CIM_NUMBERUNKNOWN. For any other value of *wAdditionalBunches*, *usBunchesRemaining* will be zero.

Comments None.

7.11 WFS_SRVE_CIM_ITEMSINSERTED

Description This service event specifies that items have been inserted into the cash-in position by the user. This event may be generated at any time.

Event Param LPWFSCIMPOSITIONINFO lpPositionInfo;

```
typedef struct _wfs_cim_position_info
     {
     WORD                         wPosition;
     WORD                         wAdditionalBunches;
     USHORT                       usBunchesRemaining;
     } WFSCIMPOSITIONINFO, *LPWFSCIMPOSITIONINFO;
```

wPosition
Specifies the position where the items have been inserted, set to one of the following values:

Value	Meaning
WFS_CIM_POSINLEFT	Items detected in the left input position.
WFS_CIM_POSINRIGHT	Items detected in the right input position.
WFS_CIM_POSINCENTER	Items detected in the center input position.
WFS_CIM_POSINTOP	Items detected in the top input position.
WFS_CIM_POSINBOTTOM	Items detected in the bottom input position.
WFS_CIM_POSINFRONT	Items detected in the front input position.
WFS_CIM_POSINREAR	Items detected in the rear input position.
WFS_CIM_POSOUTLEFT	Items detected in the left output position.
WFS_CIM_POSOUTRIGHT	Items detected in the right output position.
WFS_CIM_POSOUTCENTER	Items detected in the center output position.
WFS_CIM_POSOUTTOP	Items detected in the top output position.
WFS_CIM_POSOUTBOTTOM	Items detected in the bottom output position.
WFS_CIM_POSOUTFRONT	Items detected in the front output position.
WFS_CIM_POSOUTREAR	Items detected in the rear output position.

wAdditionalBunches
This value will always be zero within this event.

usBunchesRemaining
This value will always be zero within this event.

Comments None.

7.12 WFS_EXEE_CIM_NOTEERROR

Description This execute event specifies the reason for an item detection error during an operation which involves moving items.

Event Param LPUSHORT lpusReason;

lpusReason
Pointer to an USHORT holding the reason for the item detection error. Possible values are:

Value	Meaning
WFS_CIM_DOUBLENOTEDETECTED	Double notes have been detected.
WFS_CIM_LONGNOTEDETECTED	A long note has been detected.
WFS_CIM_SKEWEDNOTE	A skewed note has been detected.
WFS_CIM_INCORRECTCOUNT	An item counting error has occurred.
WFS_CIM_NOTESTOOCLOSE	Notes have been detected as being too close.
WFS_CIM_OTHERNOTEERROR	An item error not covered by the other values has been detected.
WFS_CIM_SHORTNOTEDETECTED	A short note has been detected.

Comments None.

7.13 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	LPWFSCIMNOTENUMBERLIST lpNoteNumberList; *lpNoteNumberList* Pointer to a WFSCIMNOTENUMBERLIST structure holding a list of banknote numbers which have been identified and accepted during execution of the sub cash-in. This field will contain the banknote numbers of the accepted items. For a description of the WFSCIMNOTENUMBERLIST structure see the definition of the WFS_INF_CIM_CASH_UNIT_INFO command.
Comments	None.

7.14 WFS_SRVE_CIM_MEDIADETECTED

Description	This service event is generated if media is detected during a reset (WFS_CMD_CIM_RESET command). 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 lpItemPosition; For a description of this parameter see the definition of the WFS_CMD_CIM_RESET command.
Comments	None.

7.15 WFS_EXEE_CIM_INPUT_P6

Description	This execute event is generated if level 2 and / or level 3 notes are detected during the cash processing operation.
Event Param	LPWFSCIMP6INFO *lppP6Info; Pointer to a NULL-terminated array of pointers to WFSCIMP6INFO structures, one structure for every level. For the description of the structure see the definition of the WFS_INF_CIM_GET_P6_INFO command.
Comments	Note: Although this event can be used to indicate that level 2 /level 3 notes have been detected, the information that it provides is limited. The more recent WFS_EXEE_CIM_INFO_AVAILABLE event combined with the WFS_INF_CIM_GET_ITEM_INFO and WFS_INF_CIM_GET_ALL_ITEM_INFO commands provide much more information. It is therefore recommended for future development that WFS_EXEE_CIM_INFO_AVAILABLE should be used in preference to this event in order to support the greatest functionality, and this event supported where backwards compatibility is necessary.
~~**Comments**~~	~~None.~~

7.16 WFS_EXEE_CIM_INFO_AVAILABLE

Description This execute event is generated when information is available for items detected during the cash processing operation.

Event Param LPWFSCIMITEMINFOSUMMARY *lppItemInfoSummary;

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

```
typedef struct _wfs_cim_item_info_summary
     {
     USHORT                    usLevel;
     USHORT                    usNumOfItems;
     } WFSCIMITEMINFOSUMMARY, *LPWFSCIMITEMINFOSUMMARY;
```

usLevel
Defines the note level. Possible values are:

Value	Meaning
WFS_CIM_LEVEL_1	Information for level 1 notes.
WFS_CIM_LEVEL_2	Information for level 2 notes.
WFS_CIM_LEVEL_3	Information for level 3 notes.
WFS_CIM_LEVEL_4	Information for level 4 notes.

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

Comments None.

7.17 WFS_EXEE_CIM_INSERTITEMS

Description	This event notifies the application when the device is ready for the user to insert items.
Event Param	None.
Comments	None.

7.18 WFS_SRVE_CIM_DEVICEPOSITION

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

Event Param LPWFSCIMDEVICEPOSITION lpDevicePosition;

```
typedef struct _wfs_cim_device_position
     {
     WORD                      wPosition;
     } WFSCIMDEVICEPOSITION, *LPWFSCIMDEVICEPOSITION;
```

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

Value	Meaning
WFS_CIM_DEVICEINPOSITION	The device is in its normal operating position.
WFS_CIM_DEVICENOTINPOSITION	The device has been removed from its normal operating position.
WFS_CIM_DEVICEPOSUNKNOWN	The position of the device cannot be determined.

Comments None.

7.19 WFS_SRVE_CIM_POWER_SAVE_CHANGE

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

Event Param LPWFSCIMPOWERSAVECHANGE lpPowerSaveChange;

```
typedef struct _wfs_cim_power_save_change
     {
     USHORT                    usPowerSaveRecoveryTime;
     } WFSCIMPOWERSAVECHANGE, *LPWFSCIMPOWERSAVECHANGE;
```

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

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

7.20 WFS_EXEE_CIM_INCOMPLETEREPLENISH

Description This execute event is generated when some items had been moved before the WFS_CMD_CIM_REPLENISH command failed with an error code (not WFS_SUCCESS), but some items were moved then the details will be reported with this event. This event can only occur once per command.

Event Param LPWFSCIMINCOMPLETEREPLENISH lpIncompleteReplenish;

```
typedef struct _wfs_cim_incomplete_replenish
     {
     LPWFSCIMREPRES            lpReplenish;
     } WFSCIMINCOMPLETEREPLENISH, *LPWFSCIMINCOMPLETEREPLENISH;
```

lpReplenish
The WFSCIMREPRES structure is defined in the description of the command WFS_CMD_CIM_REPLENISH. Note that in this case the values in this structure report the amount and number of each denomination that have actually been moved during the replenishment command.

Comments None.

7.21 WFS_EXEE_CIM_INCOMPLETEDEPLETE

Description This execute event is generated when some items had been moved before the WFS_CMD_CIM_DEPLETE command failed with an error code (not WFS_SUCCESS), but some items were moved. In this case the details will be reported with this event. This event can only occur once per command.

Event Param LPWFSCIMINCOMPLETEDEPLETE lpIncompleteDeplete;

```
typedef struct _wfs_cim_incomplete_deplete
     {
     LPWFSCIMDEPRES            lpDeplete;
     } WFSCIMINCOMPLETEDEPLETE, *LPWFSCIMINCOMPLETEDEPLETE;
```

lpDeplete
The WFSCIMDEPRES structure is defined in the description of the command WFS_CMD_CIM_DEPLETE. Note that in this case the values in this structure report the amount and number of each denomination that have actually been moved during the depletion command.

Comments None.

7.22 WFS_SRVE_CIM_SHUTTERSTATUSCHANGED

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

Event Param LPWFSCIMSHUTTERSTATUSCHANGED lpShutterStatusChanged;

```
typedef struct _wfs_cim_shutter_status_changed
     {
     WORD                      fwPosition;
     WORD                      fwShutter;
     } WFSCIMSHUTTERSTATUSCHANGED, *LPWFSCIMSHUTTERSTATUSCHANGED;
```

fwPosition
Specifies one of the CIM input or output positions whose shutter status has changed as one of the following values:

Value	Meaning
WFS_CIM_POSINLEFT	Left input position.
WFS_CIM_POSINRIGHT	Right input position.
WFS_CIM_POSINCENTER	Center input position.
WFS_CIM_POSINTOP	Top input position.
WFS_CIM_POSINBOTTOM	Bottom input position.
WFS_CIM_POSINFRONT	Front input position.
WFS_CIM_POSINREAR	Rear input position.
WFS_CIM_POSOUTLEFT	Left output position.
WFS_CIM_POSOUTRIGHT	Right output position.
WFS_CIM_POSOUTCENTER	Center output position.
WFS_CIM_POSOUTTOP	Top output position.
WFS_CIM_POSOUTBOTTOM	Bottom output position.
WFS_CIM_POSOUTFRONT	Front output position.
WFS_CIM_POSOUTREAR	Rear output position.

fwShutter
Specifies the new state of the shutter as one of the following values:

Value	Meaning
WFS_CIM_SHTCLOSED	The shutter is closed.
WFS_CIM_SHTOPEN	The shutter is opened.
WFS_CIM_SHTJAMMED	The shutter is jammed.
WFS_CIM_SHTUNKNOWN	Due to a hardware error or other condition, the state of the shutter cannot be determined.

Comments None.

7.23 WFS_SRVE_CIM_COUNTACCURACYCHANGED

Description	This service event is generated when information about the accuracy of *ulCount* contained in the logical or physical cash unit is changed.
Event Param	LPWFSCIMCASHUNITCOUNTSTATUS lpCashUnitCountStatus;
	For the description of the structure see the definition of the WFS_INF_CIM_CASH_UNIT_COUNT_STATUS command.
Comments	None.

8. ATM Cash-In Transaction Flow - Application Guidelines

The following table is a summary of the application flows required given the possible values for *bShutterControl* and *bItemsTakenSensor* for a successful cash-in transaction. In all cases *bPresentControl* == TRUE.

	bItemsInsertedSensor == TRUE	*bItemsInsertedSensor* == FALSE
bShutterControl == TRUE	WFS_CMD_CIM_CASH_IN_START	WFS_CMD_CIM_CASH_IN_START
	WFS_CMD_CIM_CASH_IN	WFS_CMD_CIM_CASH_IN
	InsertedEvent generated	
	WFS_CMD_CIM_CASH_IN_END	WFS_CMD_CIM_CASH_IN_END
bShutterControl == FALSE	WFS_CMD_CIM_CASH_IN_START	WFS_CMD_CIM_CASH_IN_START
	WFS_CMD_CIM_OPEN_SHUTTER	WFS_CMD_CIM_OPEN_SHUTTER
	InsertedEvent generated	User Input
	WFS_CMD_CIM_CLOSE_SHUTTER	WFS_CMD_CIM_CLOSE_SHUTTER
	WFS_CMD_CIM_CASH_IN	WFS_CMD_CIM_CASH_IN
	WFS_CMD_CIM_CASH_IN_END	WFS_CMD_CIM_CASH_IN_END

The following sections describe the flow of a cash-in transaction on a Self-Service CIM. These application flows are provided as guidelines only.

8.1 OK Transaction (Explicit Shutter Control)

The following table describes a normal cash-in transaction flow where everything works and the shutter is explicitly controlled by the application.

This flow covers the following cases:

- *bShutterControl* == FALSE, *bItemsInsertedSensor* == TRUE
- *bShutterControl* == FALSE, *bItemsInsertedSensor* == FALSE

Step	Customer	Application	XFS Commands and Events
1.	Customer selects cash-in operation.		WFS_CMD_CIM_CASH_IN_START
2.		Open the shutter of the input tray.	WFS_CMD_CIM_OPEN_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_EXEE_CIM_INSERTITEMS
3.		Ask the customer to insert money.	
4.	Customer inserts money.		
5.	If *bItemsInsertedSensor* == FALSE, confirm completion.		If *bItemsInsertedSensor* == TRUE: WFS_SRVE_CIM_ITEMSINSERTED
6.		Close shutter.	WFS_CMD_CIM_CLOSE_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
7.			* WFS_CMD_CIM_CASH_IN initiated The bill recognition begins.
8.			* WFS_CMD_CIM_CASH_IN completion
9.		Display the number of items and/or amount recognized so far.	
10.		Ask the customer for further actions: If the customer wants to insert more money: Repeat from step 2. If the customer wants to finish the transaction: Continue with step 11. If the customer wants to get back all items inserted so far see table "Cancellation by Customer (Explicit Shutter Control)"	
11.		Transport the money into the cash units ~~RECYCLE_UNIT/CASHINBOX~~of type WFS_CIM_TYPERECYCLING / WFS_CIM_TYPECASHIN.	WFS_CMD_CIM_CASH_IN_END
12.		Credit the money to the customer's account.	
13.		End of transaction.	

8.2 Cancellation by Customer (Explicit Shutter Control)

The following table describes the flow of a cash-in transaction where the customer wants all the items to be returned after recognition.

This flow covers the following cases:

- *bShutterControl* == FALSE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE
- *bShutterControl* == FALSE, *bItemsInsertedSensor* == FALSE, *bItemsTakenSensor* == TRUE
- *bShutterControl* == FALSE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == FALSE
- *bShutterControl* == FALSE, *bItemsInsertedSensor* == FALSE, *bItemsTakenSensor* == FALSE

Step	Customer	Application	XFS Commands and Events
1.- 10.	See OK Transaction (Explicit Shutter Control).		
11.	Selection: Return all the items.		
12.		Transport the items recognized to the output position.	WFS_CMD_CIM_CASH_IN_ROLLBACK
13.		Open shutter.	WFS_CMD_CIM_OPEN_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED
14.		Request removal of the money.	
15.	Customer takes the money from the output position.		
16.	If *bItemsTakenSensor* == FALSE, confirm completion or use application timeout.		If *bItemsTakenSensor* == TRUE: WFS_SRVE_CIM_ITEMSTAKEN
17.		Close shutter.	WFS_CMD_CIM_CLOSE_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
18.		End of transaction.	

8.3 Stacker Becomes Full (Explicit Shutter Control)

The following table describes the flow of a cash-in transaction when the stacker becomes full during the transaction and the shutter is explicitly controlled by the application. This flow covers the following cases:

- *bShutterControl* == FALSE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE
- *bShutterControl* == FALSE, *bItemsInsertedSensor* == FALSE, *bItemsTakenSensor* == TRUE
- *bShutterControl* == FALSE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == FALSE
- *bShutterControl* == FALSE, *bItemsInsertedSensor* == FALSE, *bItemsTakenSensor* == FALSE

Step	Customer	Application	XFS Commands and Events
1.-6.	See OK Transaction (Explicit Shutter Control).		
7.			* WFS_CMD_CIM_CASH_IN initiated. The bill recognition begins.
8.			WFS_EXEE_CIM_INPUTREFUSE (WFS_CIM_STACKERFULL) … * WFS_CMD_CIM_CASH_IN completes with WFS_SUCCESS
9.		Open shutter.	WFS_CMD_CIM_OPEN_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED
10.		Ask the customer to remove the excess items.	
11.	Customer removes excess money.		
12.	If *bItemsTakenSensor* == FALSE: confirm completion or use application timeout.		If *bItemsTakenSensor* == TRUE: WFS_SRVE_CIM_ITEMSTAKEN
13.		Close shutter	WFS_CMD_CIM_CLOSE_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
14.		Display the amount recognized so far and tell the customer that the stacker is full.	
15.		Ask the customer for further actions: If the customer wants to deposit the amount: Continue with step 16. If the customer wants to get back all items inserted so far see table "Cancellation by Customer (Explicit Shutter Control)"	
16.		Transport the money into the cash units ~~RECYCLE_UNIT/CASHINBOX~~of type WFS_CIM_TYPERECYCLING / WFS_CIM_TYPECASHIN.	WFS_CMD_CIM_CASH_IN_END

17.		Ask the customer if the customer wants to deposit more money. If the customer wants to deposit more: Repeat from step 1. If the customer wants to finish the transaction: Continue with step 18.	
18.		Credit the money to the customer's account.	
19.		End of transaction.	

8.4 Bill Recognition Error (Explicit Shutter Control)

The following table describes the flow of a cash-in transaction when the items are rejected as unrecognized during the transaction and the shutter is explicitly controlled by the application.

This flow covers the following cases:

- *bShutterControl* == FALSE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE
- *bShutterControl* == FALSE, *bItemsInsertedSensor* == FALSE, *bItemsTakenSensor* == TRUE
- *bShutterControl* == FALSE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == FALSE
- *bShutterControl* == FALSE, *bItemsInsertedSensor* == FALSE, *bItemsTakenSensor* == FALSE

Step	Customer	Application	XFS Commands and Events
1.-6.	See OK Transaction (Explicit Shutter Control).		
7.			* WFS_CMD_CIM_CASH_IN initiated. The bill recognition begins.
8.			WFS_EXEE_CIM_INPUTREFUSE (WFS_CIM_INVALIDBILL) … * WFS_CMD_CIM_CASH_IN completes with WFS_SUCCESS
9.		Open shutter.	WFS_CMD_CIM_OPEN_SHUTTER WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPENED) WFS_SRVE_CIM_ITEMSPRESENTED
10.		Tell the customer that the items were not recognized and that the customer should take the items.	
11.	Customer removes unrecognized money		
12.	If *bItemsTakenSensor* == FALSE: confirm completion or use application timeout.		If *bItemsTakenSensor* == TRUE: WFS_SRVE_CIM_ITEMSTAKEN
13.		Close shutter.	WFS_CMD_CIM_CLOSE_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
14.		Display the amount recognized so far.	
15.		Ask the customer for further actions: If the customer wants to deposit the amount: Continue with step 16. If the customer wants to get back all items inserted so far see table "Cancellation by Customer (Explicit Shutter Control)"	

16.		Transport the money into the cash units ~~RECYCLE_UNIT/CASHINBOX~~of type WFS_CIM_TYPERECYCLING / WFS_CIM_TYPECASHIN.	WFS_CMD_CIM_CASH_IN_END
17.		Credit the money to the customer's account.	
18.		End of transaction.	

8.5 OK Transaction (Explicit Shutter Control) - Level 2 and 3 Note ~~Handling Standard~~classification Supported

This section describes a possible cash-in transaction where ~~a note handing standard~~Level 2 and 3 Note classification is supported and everything works fine when level 2 / level 3 notes are inserted.

This flow covers the following cases:

- *bShutterControl* == FALSE, *bItemsInsertedSensor* == TRUE

Step	Customer	Application	XFS Command
1.	Select function cash-in.	Open the shutter of the input tray.	WFS_CMD_CIM_CASH_IN_START WFS_CMD_CIM_OPEN_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_EXEE_CIM_INSERTITEMS
2.		Ask the customer to insert money.	
3.	Customer inserts money.		WFS_SRVE_CIM_ITEMSINSERTED WFS_CMD_CIM_CLOSE_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED) * WFS_CMD_CIM_CASH_IN initiated The bill recognition begins.
4.			WFS_EXEE_CIM_INPUTP6 * WFS_CMD_CIM_CASH_IN completes
5.		Get number of level 2 / level 3 notes.	WFS_INF_CIM_GET_P6_INFO
6.		Display the amount recognized so far and inform customer that level 2 / level 3 notes are inserted.	
7.		Store signatures of level 2 / level 3 notes with customer data.	Call command WFS_INF_CIM_GET_P6_SIGNATURE once for every signature.
8.		Ask the customer for further actions: If the customer wants to insert more money: Repeat from step 2. If the customer wants to finish the transaction: Continue with step 9. If the customer wants to get back all items inserted so far see table "cancellation by customer"	
9.		Transport the money into the cash units ~~RECYCLE_UNIT/CASHINBOX~~of type WFS_CIM_TYPERECYCLING / WFS_CIM_TYPECASHIN.	WFS_CMD_CIM_CASH_IN_END

10.		At this point the application should decide how to credit the appropriate money to the customer's account, and inform the customer about the amounts of level 2 and level 3 notes.	
11.		End of transaction.	

8.6 Multiple Bunches Returned During WFS_CMD_CIM_CASH_IN Refused Notes (Explicit Shutter Control)

The following table describes the flow of a cash-in transaction where items are rejected during the transaction and the Service Provider has explicit shutter control. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are used. Additionally, the number of items refused may be greater than the number of items that can be presented at the output position.

This flow covers the following cases:

- *bShutterControl* == FALSE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE, *bPresentControl* == FALSE

Step	**Customer**	**Application**	**XFS Commands and Events**
1.-6.	See OK Transaction (Explicit Shutter Control).		
7.			* WFS_CMD_CIM_CASH_IN initiated. The bill recognition begins.
8.			WFS_EXEE_CIM_INPUTREFUSE (WFS_CIM_INVALIDBILL) … * WFS_CMD_CIM_CASH_IN completes with WFS_SUCCESS
9.		Open shutter.	WFS_CMD_CIM_OPEN_SHUTTER WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPENED) WFS_SRVE_CIM_ITEMSPRESENTED WFS_CMD_CIM_OPEN_SHUTTER completes with WFS_SUCCESS
10.		If there are additional bunches to deliver then this can be determined from the output parameter of the WFS_SRVE_CIM_ITEMSPRESENTED event. Tell the customer that the items were not accepted, and to take the items. The customer should be informed that the items will be returned in multiple bunches.	
11.	Customer takes the bunch of items.		WFS_SRVE_CIM_ITEMSTAKEN
12.		Close shutter.	WFS_CMD_CIM_CLOSE_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED) WFS_CMD_CIM_CLOSE_SHUTTER completes with WFS_SUCCESS

13.		Check if more refused bills need to be taken. The *wAdditionalBunches* and *usBunchesRemaining* fields from the last WFS_SRVE_CIM_ITEMSPRESENTED event are used to determine this. Note that if more items are to be presented, the WFS_CMD_CIM_OPEN_SHUTTER in step 9 will move the next bunch to the output position. If *wAdditionalBunches* == WFS_CIM_ADDBUNCHONEMORE Repeat steps 9. – 13. Else Go to step 14.	
14.		Display the amount recognized so far.	
15.		Ask the customer for further actions: If the customer wants to deposit the amount: Continue with step 16. If the customer wants to get back all items inserted so far see table "Multiple Bunches Returned During WFS_CMD_CIM_CASH_IN_ROLLBACK"	
16.		Transport the money into the cash units of type WFS_CIM_TYPERECYCLING / WFS_CIM_TYPECASHIN.	WFS_CMD_CIM_CASH_IN_END
17.		Credit the money to the customer's account.	
18.		End of transaction.	

8.7 Multiple Bunches Returned During WFS_CMD_CIM_CASH_IN_ROLLBACK (Explicit Shutter Control)

The following table describes the flow of a roll back operation where items are rolled back during the transaction and the Service Provider has explicit shutter control. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are used. Additionally, the number of items rolled back may be greater than the number of items that can be presented at the output position.

This flow covers the following cases:

- *bShutterControl* == FALSE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE, *bPresentControl* == FALSE

Step	**Customer**	**Application**	**XFS Commands and Events**
1.-10.	See OK Transaction (Explicit Shutter Control).		
11.	Selection: Return all the items.		
12.		Transport the items recognized to the output position.	WFS_CMD_CIM_CASH_IN_ROLLBACK
			WFS_CMD_CIM_CASH_IN_ROLLBACK completes with WFS_SUCCESS.
13.		Open shutter.	WFS_CMD_CIM_OPEN_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED WFS_CMD_CIM_OPEN_SHUTTER completes with WFS_SUCCESS
14.		Tell the customer to take the items. The customer should be informed that the items will be returned in multiple bunches. If there are additional bunches to deliver then this can be determined from the output parameter of the WFS_SRVE_CIM_ITEMSPRESENTED event.	
15.	Customer takes the bunch of items.		WFS_SRVE_CIM_ITEMSTAKEN
16.		Close shutter.	WFS_CMD_CIM_CLOSE_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED) WFS_CMD_CIM_CLOSE_SHUTTER completes with WFS_SUCCESS

17.		Check if more bills need to be taken. The *wAdditionalBunches* and *usBunchesRemaining* fields from the last WFS_SRVE_CIM_ITEMSPRESENTED event is used to determine this. Note that if more items are to be presented, the WFS_CMD_CIM_OPEN_SHUTTER in step 13 will move the next bunch to the output position. If *wAdditionalBunches* == WFS_CIM_ADDBUNCHONEMORE Repeat steps 13. – 17. Else Go to step 18.	
18.		End of transaction.	

~~8.6~~8.8 OK Transaction (Implicit Shutter Control)

The following table describes a normal cash-in transaction flow where everything works and the shutter is implicitly controlled by the Service Provider. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are not explicitly used by the application.

This flow covers the following cases:

- *bShutterControl* == TRUE, *bItemsInsertedSensor* == TRUE
- *bShutterControl* == TRUE, *bItemsInsertedSensor* == FALSE

Step	Customer	Application	XFS Commands and Events
1.	Customer selects cash-in operation.		WFS_CMD_CIM_CASH_IN_START
2.			* WFS_CMD_CIM_CASH_IN initiated The Service Provider implicitly opens the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_EXEE_CIM_INSERTITEMS event is sent when the shutter is fully open and the device is ready to begin accepting items.
3.		Ask the customer to insert money.	
4.	Customer inserts money.		
5.			If *bItemsInsertedSensor* == TRUE: WFS_SRVE_CIM_ITEMSINSERTED The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED) The bill recognition begins.
6.			* WFS_CMD_CIM_CASH_IN command completes.
7.		Display the number of items and/or amount recognized so far.	
8.		Ask the customer for further actions: If the customer wants to insert more money: Repeat from step 2. If the customer wants to finish the transaction: Continue with step 9. If the customer wants to get back all items inserted so far see table "Cancellation by Customer (Implicit Shutter Control)"	
9.	Selection: Finish the transaction		

10.		Transport the money into the cash units ~~RECYCLE_UNIT/CASHINBOX~~of type WFS_CIM_TYPERECYCLING / WFS_CIM_TYPECASHIN.	WFS_CMD_CIM_CASH_IN_END
11.		Credit the money to the customer's account.	
12.		End of transaction.	

~~8.7~~8.9 ~~Cancellation by~~ Customer Initiates Returning Of Previously Recognized Items (Implicit Shutter Control)

The following table describes the flow of a cash-in transaction where the customer wants all the items to be returned after recognition and the shutter is implicitly controlled by the Service Provider. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are not used.

This flow covers the following cases:

- *bShutterControl* == TRUE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE
- *bShutterControl* == TRUE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == FALSE

Step	Customer	Application	XFS Commands and Events
1.-8.	See OK Transaction (Implicit Shutter Control).		
9.	Selection: Return all the items.		
10.		Transport the items recognized to the output position.	WFS_CMD_CIM_CASH_IN_ROLLBACK The Service Provider implicitly opens the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED
11.		Request removal of the money.	
12.	Customer takes the money from the output position.		
13.	If *bItemsTakenSensor* == FALSE: confirm completion or use application timeout.		If *bItemsTakenSensor* == TRUE: WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
14.		End of transaction	

~~8.8~~8.10 OK Transaction - (Implicit Shutter Control and WFS_EXEE_CIM_SUBCASHIN event supported)

The following table describes the chronological steps taken in the flow of a cash-in transaction where the cash-in operation is subdivided into a number of logical operations under hardware control. In this case a WFS_EXEE_CIM_SUBCASHIN event is generated for each sub cash-in operation. This may be the case for instance where a device does its coin or bill recognition in batches of 25. In this case the Service Provider would post a WFS_EXEE_CIM_SUBCASHIN event each time 25 items were processed. In this example the shutter is implicitly controlled by the Service Provider so the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are not used.

This flow covers the following cases:

- *bShutterControl* == TRUE, *bItemsInsertedSensor* == TRUE
- *bShutterControl* == TRUE, *bItemsInsertedSensor* == FALSE

Step	Customer	Application	XFS Commands and Events
1.-5.	See OK Transaction (Implicit Shutter Control).		
6.			The device processes the items in batches. Each time a batch is completed a WFS_EXEE_CIM_SUBCASHIN event is posted then the cash-in operation continues.
7.			* WFS_CMD_CIM_CASH_IN completes.
8.		Display the number of items and/or amount recognized so far.	
9.		Ask the customer for further actions: If the customer wants to insert more money: Repeat from step 2. If the customer wants to finish the transaction: Continue with step 10. If the customer wants to get back all items inserted so far see table "Cancellation by Customer (Implicit Shutter Control)"	
10.			WFS_CMD_CIM_CASH_IN_END
11.		End of transaction.	

~~8.9~~8.11 Multiple ~~Refused Notes~~Bunches Returned During WFS_CMD_CIM_CASH_IN (Implicit Shutter Control~~)~~ and Implicit Present Control)

The following table describes the flow of a cash-in transaction where items are rejected during the transaction and the Service Provider ~~implicitly controls the~~has implicit shutter~~.~~ and present control. In this case the WFS_CMD_CIM_OPEN_SHUTTER ~~and~~, WFS_CMD_CIM_CLOSE_SHUTTER and WFS_CMD_CIM_PRESENT_MEDIA commands are not used. Additionally, the number of items refused may be greater than the number of items that can be presented at the output position. Due to the complexity of this scenario, ~~control of the~~ shutter and present control must be implicit. Therefore, there is no corresponding flow for explicit shutter and present control.

~~Step~~	~~Customer~~	~~Application~~	~~XFS Command~~

This flow covers the following cases:

- *bShutterControl* == TRUE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE, *bPresentControl* == TRUE

Step	Customer	Application	XFS Command
1.-5.	See OK Transaction (Implicit Shutter Control).		
~~6.~~			~~As a result of the bill processing n batches of bills must be returned to the customer.~~
6.			As a result of the bill processing n bunches of items must be returned to the customer.
7.			WFS_EXEE_CIM_INPUTREFUSE
8.			Return ~~batch~~bunch 1 of ~~bills~~items to customer. The Service Provider implicitly opens the shutter and implicitly presents the bunch of items. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED
9.		Tell the customer that the ~~bills~~items were not accepted, and to take the ~~bills~~items. The customer should be informed that the items will be returned in multiple bunches. If there are additional bunches to deliver then this can be determined from the output parameter of the WFS_SRVE_CIM_ITEMSPRESENTED event.	
10.	Customer ~~removes unrecognized money~~takes the bunch of items.		WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
11.			Return bunch 2 of items to customer. The Service Provider implicitly opens the shutter and implicitly presents the bunch of items. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED

12.		~~Tell the customer to take the bills.~~Tell the customer that the items were not accepted, and to take the items. The customer should be informed that the items will be returned in multiple bunches. If there are additional bunches to deliver then this can be determined from the output parameter of the WFS_SRVE_CIM_ITEMSPRESENTED event.	
13.	Customer ~~removes unrecognized money~~takes the bunch of items.		WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
14.			Repeat steps 11.-13. until ~~batches 2~~bunches 3 to n-1 are returned to the customer.
15.			Return ~~Batch~~bunch n (last) of ~~notes~~items to customer. The Service Provider implicitly opens the shutter and implicitly presents the bunch of items. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_-SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED
16.			* WFS_CMD_CIM_CASH_IN completes with WFS_SUCCESS.
17.		Tell the customer to take the ~~bills~~items. The customer should be informed that this is the final bunch.	
18.	Customer ~~removes unrecognized money~~takes the bunch of items.		
19.			WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
20.		Display the amount recognized so far.	
21.		Ask the customer for further actions: If the customer wants to deposit the amount: Continue with step 21. If the customer wants to get back all items inserted so far see table "Cancellation by Customer (Implicit Shutter Control)"	

22.		Transport the money into the cash units ~~RECYCLE_UNIT/CASHINBOX~~of type WFS_CIM_TYPERECYCLING / WFS_CIM_TYPECASHIN.	WFS_CMD_CIM_CASH_IN_END
23.		Credit the money to the customer's account.	
24.		End of transaction.	

~~8.10~~8.12 Multiple ~~Rollback Notes~~Bunches Returned During WFS_CMD_CIM_CASH_IN_ROLLBACK (Implicit Shutter Control~~)~~ and Implicit Present Control)

The following table describes the flow of a roll back operation where items are rolled back during the transaction and the Service Provider ~~implicitly controls the~~has implicit shutter~~.~~ and present control. In this case the WFS_CMD_CIM_OPEN_SHUTTER ~~and~~, WFS_CMD_CIM_CLOSE_SHUTTER and WFS_CMD_CIM_PRESENT_MEDIA commands are not used. Additionally, the number of items rolled back may be greater than the number of items that can be presented at the output position. Due to the complexity of this scenario, ~~control of the~~ shutter and present control must be implicit. Therefore, there is no corresponding flow for explicit shutter and present control.

This flow covers the following cases:

- *bShutterControl* == TRUE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE, *bPresentControl* == TRUE

Step	Customer	Application	XFS Command

~~Step~~	~~Customer~~	~~Application~~	~~XFS Command~~
~~1.-9.~~	~~See Cancellation by Customer (Implicit Shutter Control).~~		
1.-9.	See Customer Initiates Returning Of Previously Recognized Items (Implicit Shutter Control).		
10.		Initiate the roll back operation.	* WFS_CMD_CIM_CASH_IN_ROLLBACK
11.			The Service Provider begins the roll back. As a result of this n ~~batches~~bunches of ~~notes~~items must be returned to the customer.
12.			Return ~~batch~~bunch 1 of ~~notes~~items to customer. The Service Provider implicitly opens the shutter and implicitly presents the bunch of items. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED
13.		~~Tell the customer to take the bills.~~Tell the customer to take the items. The customer should be informed that the items will be returned in multiple bunches. If there are additional bunches to deliver then this can be determined from the output parameter of the WFS_SRVE_CIM_ITEMSPRESENTED event.	
14.	Customer ~~removes money~~takes the bunch of items.		WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)

15.			Repeat steps 11.-14. until ~~batches~~bunches 2 to n-1 are returned to the customer.
16.			Return ~~batch~~bunch n (last) of ~~notes~~items to customer. The Service Provider implicitly opens the shutter and implicitly presents the bunch of items. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED
17.			* WFS_CMD_CIM_CASH_IN_ROLLBACK completes with WFS_SUCCESS.
18.		Tell the customer to take the ~~bills~~items. The customer should be informed that this is the final bunch.	
19.	Customer ~~removes money~~takes the bunch of items.		
20.			WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
21.		End of transaction.	

8.13 Retracting Items When Multiple Bunches Are Returned During WFS_CMD_CIM_CASH_IN (Implicit Shutter Control and Implicit Present Control)

The following table describes the flow of a cash-in transaction where items are returned back during the transaction and the Service Provider has implicit shutter and present control. In this case the WFS_CMD_CIM_OPEN_SHUTTER, WFS_CMD_CIM_CLOSE_SHUTTER and WFS_CMD_CIM_PRESENT_MEDIA commands are not used. Additionally, the number of items returned may be greater than the number of items that can be presented at the output position. Due to the complexity of this scenario, shutter and present control must be implicit. Therefore, there is no corresponding flow for explicit shutter and present control.

This flow covers the following cases:

- *bShutterControl* == TRUE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE, *bPresentControl* == TRUE

Step	**Customer**	**Application**	**XFS Command**
1.-5.	See OK Transaction (Implicit Shutter Control).		
6.			As a result of the bill processing n bunches of items must be returned to the customer.
7.			WFS_EXEE_CIM_INPUTREFUSE
8.			Return bunch 1 of items to customer. The Service Provider implicitly opens the shutter and implicitly presents the bunch of items. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED
9.		Tell the customer that the items were not accepted, and to take the items. The customer should be informed that the items will be returned in multiple bunches. If there are additional bunches to deliver then this can be determined from the output parameter of the WFS_SRVE_CIM_ITEMSPRESENTED event.	
10.	Customer **does not** take the bunch of items.		
11.		After some time the application timeout waiting for the items to be taken is reached	WFSCancelAsyncRequest is executed to end the WFS_CMD_CIM_CASH_IN command.
12.			* If command cancellation is supported the WFS_CMD_CIM_CASH_IN completes with WFS_ERR_CANCELED.
13.		All items are retracted.	WFS_CMD_CIM_RETRACT
14.		End of transaction.	

~~8.11~~8.14 Bill Recognition Error (WFS_CMD_CIM_PRESENT_MEDIA Command Supported)

The following table describes the flow of a cash-in transaction when the items are rejected as unrecognized during the transaction and the WFS_CMD_CIM_PRESENT_MEDIA command is supported.

This flow covers the following case:

- *bShutterControl* == FALSE, *bPresentControl* == FALSE, *bItemsTakenSensor* == TRUE

Step	Customer	Application	XFS Commands and Events
1.-7.	See OK Transaction (Explicit Shutter Control).		
8.			WFS_EXEE_CIM_INPUTREFUSE (WFS_CIM_INVALIDBILL) * WFS_CMD_CIM_CASH_IN completes with WFS_SUCCESS.
9.		Present items to customer.	* WFS_CMD_CIM_PRESENT_MEDIA initiated. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED
10.			* WFS_CMD_CIM_PRESENT_MEDIA completes
11.		Tell the customer that the items were not recognized and that the customer should take the items.	
12.	Customer removes unrecognized money.		
13.			WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
14.		Display the amount recognized so far.	
15.		Ask the customer for further actions: If the customer wants to deposit the amount: Continue with step 15. If the customer wants to get back all items inserted so far see table "Cancellation by Customer (Explicit Shutter Control)"	
16.		Transport the money into the cash units ~~RECYCLE_UNIT/CASHINBOX~~of type WFS_CIM_TYPERECYCLING / WFS_CIM_TYPECASHIN.	WFS_CMD_CIM_CASH_IN_END
17.		Credit the money to the customer's account.	
18.		End of transaction.	

~~8.12~~8.15 Cancellation by Customer (Implicit Shutter Control and WFS_CMD_CIM_PRESENT_MEDIA Command Supported)

The following table describes the flow of a cash-in transaction where the customer wants all the items to be returned after recognition and the WFS_CMD_CIM_PRESENT_MEDIA command is supported.

This flow covers the following case:

- *bShutterControl* == TRUE, *bPresentControl* == FALSE, *bItemsTakenSensor* == TRUE

Step	**Customer**	**Application**	**XFS Commands and Events**
1.-9.	See Cancellation by Customer (Implicit Shutter Control).		
10.		Transport the items recognized to an internal position.	* WFS_CMD_CIM_CASH_IN_ROLLBACK initiated
11.			* WFS_CMD_CIM_CASH_IN_ROLLBACK completes.
12.		Present items to the customer.	* WFS_CMD_CIM_PRESENT_MEDIA initiated. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED
13.			* WFS_CMD_CIM_PRESENT_MEDIA completes.
14.		Request removal of the money.	
15.	Customer takes the money from the output position.		
16.			WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
17.		End of transaction.	

8.16 Multiple Bunch Timeout Handling

The following sections describe flows where the Service Provider could potentially present refused items in multiple bunches during the WFS_CMD_CIM_CASH_IN command. As the WFS_CMD_CIM_CASH_IN timeout (*dwTimeout* parameter in WFSAsyncExecute or WFSExecute) may elapse before the last bunch is presented, resulting in a WFS_ERR_TIMEOUT in the completion event, it is recommended that the application take control by specifying a long *dwTimeout* and use timers to allow sufficient time for user interaction before cancelling the command. *dwTimeout* should be set sufficiently long to allow for any scenario; it could be set to WFS_INDEFINITE_WAIT as the command would be explicitly cancelled by the application if timers elapse.

Each flow covers the following cases:

- *bShutterControl* == TRUE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE, *bPresentControl* == TRUE

8.16.1 No Items Inserted

In this flow, the user does not insert items within the required time, therefore the application cancels the WFS_CMD_CIM_CASH_IN command using WFS_CMD_CIM_CASH_IN_END.

Step	Customer	Application	XFS Command
1.	Customer selects cash-in operation.		WFS_CMD_CIM_CASH_IN_START
2.			* WFS_CMD_CIM_CASH_IN initiated with a long timeout (for example, WFS_INDEFINITE_WAIT) using WFSAsyncExecute The Service Provider implicitly opens the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_EXEE_CIM_INSERTITEMS event is sent when the shutter is fully open and the device is ready to begin accepting items.
3.		Ask the customer to insert money. Application sets an insertion timer.	
4.	Customer does not insert money.		
5.		The insertion timer elapses	WFSCancelAsyncRequest is executed to end the WFS_CMD_CIM_CASH_IN command.
6.			* If command cancellation is supported the WFS_CMD_CIM_CASH_IN completes with WFS_ERR_CANCELED. WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
7.		Transaction cancelled	WFS_CMD_CIM_CASH_IN_END
8.		End of transaction.	

8.16.2 First Bunch Not Taken

In this flow, the user does not take the first returned bunch within the required time, therefore the application cancels the WFS_CMD_CIM_CASH_IN command. The same sequence can be extended to any bunch other than the last bunch as this would complete the WFS_CMD_CIM_CASH_IN command; each time a new bunch is presented a new presentation timer should be set.

Step	Customer	Application	XFS Commands and Events
1.-3.	See No Items Inserted		
4.	Customer inserts money		If *bItemsInsertedSensor* == TRUE: WFS_SRVE_CIM_ITEMSINSERTED The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED) The bill recognition begins.
5.		Insertion timer cancelled	
6.			As a result of the bill processing n bunches of items must be returned to the customer.
7.			WFS_EXEE_CIM_INPUTREFUSE
8.			Return bunch 1 of items to customer. The Service Provider implicitly opens the shutter and implicitly presents the bunch of items. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED
9.		Tell the customer that the items were not accepted, and to take the items. The customer should be informed that the items will be returned in multiple bunches. If there are additional bunches to deliver then this can be determined from the output parameter of the WFS_SRVE_CIM_ITEMSPRESENTED event. Presentation timer set	
10.	Customer does not take the items	The presentation timer elapses	WFSCancelAsyncRequest is executed to end the WFS_CMD_CIM_CASH_IN command.
			* If command cancellation is supported the WFS_CMD_CIM_CASH_IN completes with WFS_ERR_CANCELED. WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
11.		All items are retracted.	WFS_CMD_CIM_RETRACT
12.		End of transaction.	

8.16.3 Last Bunch Taken

In this flow, two bunches are to be returned & the user takes all of the returned bunches within the required time, therefore WFS_CMD_CIM_CASH_IN command completes normally.

Step	Customer	Application	XFS Commands and Events
1.-9.	See First Bunch Not Taken		

10.	Customer takes the bunch		WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
11.		Presentation timer cancelled	Return bunch 2 of items to customer. The Service Provider implicitly opens the shutter and implicitly presents the bunch of items. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED
12.			* WFS_CMD_CIM_CASH_IN completes with WFS_SUCCESS.
13.	Customer takes the bunch of items.		
14.			WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED)
15.		Display the amount recognized so far.	
16.		Ask the customer for further actions: If the customer wants to deposit the amount: Continue with step 17. If the customer wants to get back all items inserted so far see table "Cancellation by Customer (Implicit Shutter Control)"	
17.		Transport the money into the cash units of type WFS_CIM_TYPERECYCLING / WFS_CIM_TYPECASHIN.	WFS_CMD_CIM_CASH_IN_END
18.		Credit the money to the customer's account.	
19.		End of transaction.	

8.17 Exchange using DEPOSITINTO (Implicit Shutter Control)

The following table describes an Exchange using the WFS_CIM_DEPOSITINTO parameter to specify that items will be deposited using the deposit entrance. The shutter is implicitly controlled by the Service Provider. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are not explicitly used by the application.

Although this re-uses Cash In transaction commands to move the items, the Exchange is not restricted by the maximum number of items in a Cash In transaction (*fwIntermediateStacker*) as the Exchange can be performed using multiple deposits. Items may be returned or captured per local policy and configuration. Despite using the standard Cash In transaction commands, this sequence does not constitute one or more Cash In transactions therefore is not reported by WFS_INF_CIM_CASH_IN_STATUS. Other Cash In transaction commands such as WFS_CMD_CIM_CASH_IN_ROLLBACK can be used if required. Note also that in this example flow, each bunch will be transported to cash units before additional items can be inserted; it is equally valid to accept multiple bunches before depositing the items to the cash units.

This example flow covers cases where all the items are accepted during WFS_CMD_CIM_CASH_IN; unrecognized items may be deposited to a cash unit with the *fwItemType* containing WFS_CIM_CITYPLEVEL1. Refer to other example flows for how refused items would be handled.

This flow covers the following case:

- *bShutterControl* == TRUE, *bItemsInsertedSensor* == TRUE, *fwIntermediateStacker* != 0

Step	**User**	**Application**	**XFS Commands and Events**
1.	User selects to perform a replenishment using the deposit entrance.		WFS_CMD_CIM_START_EXCHANGE with *fwExchangeType* == WFS_CIM_DEPOSITINTO.
2.			WFS_CMD_CIM_CASH_IN_START called to specify the input position.
3.			WFS_CMD_CIM_CASH_IN initiated The Service Provider implicitly opens the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_EXEE_CIM_INSERTITEMS event is sent when the shutter is fully open and the device is ready to begin accepting items.
4.		Ask the user to insert items.	
5.	User inserts items.		
6.			WFS_SRVE_CIM_ITEMSINSERTED The Service Provider implicitly closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED) The bill recognition begins.
7.			WFS_CMD_CIM_CASH_IN command completes.
8.		Display the number of items and/or amount recognized so far.	
8.		Transport the items into the designated cash units.	WFS_CMD_CIM_CASH_IN_END

9.		Ask the user for further actions: If the user wants to insert more items: Repeat from step 2. If the user wants to complete the Exchange operation: Continue with step 10.	
10.	Selection: Complete		
11.			WFS_CMD_CIM_END_EXCHANGE. This can be specified with a NULL input parameter as all the notes will have been counted and cash unit counts adjusted accordingly during the preceding operations.
12.		End of Exchange.	

8.18 Multiple Bunches Returned During WFS_CMD_CIM_CASH_IN Refused Notes (using WFS_CMD_CIM_PREPARE_PRESENT)

The following table describes the flow of a cash-in transaction where items are rejected during the transaction. The application uses WFS_CMD_CIM_PREPARE_PRESENT commands to move items to the output position. The Service Provider has explicit shutter control. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are used for the user to take items. Additionally, the number of items refused may be greater than the number of items that can be presented at the output position.

This flow covers the following cases:

- *bShutterControl* == FALSE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE, *bPresentControl* == FALSE, *bPreparePresent* == TRUE

Step	Customer	Application	XFS Commands and Events
1.-6.	See OK Transaction (Explicit Shutter Control).		
7.			* WFS_CMD_CIM_CASH_IN initiated. The bill recognition begins. * WFS_CMD_CIM_CASH_IN resets the *lpTotalReturnedItems* output parameter of WFS_INF_CIM_PRESENT_STATUS.
8.			WFS_EXEE_CIM_INPUTREFUSE (WFS_CIM_INVALIDBILL) … * WFS_CMD_CIM_CASH_IN completes with WFS_SUCCESS
9.		Move refused items to the output position.	WFS_CMD_CIM_PREPARE_PRESENT WFS_CMD_CIM_PREPARE_PRESENT completes with WFS_SUCCESS
10.		Open shutter.	WFS_CMD_CIM_OPEN_SHUTTER WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPENED) WFS_SRVE_CIM_ITEMSPRESENTED WFS_CMD_CIM_OPEN_SHUTTER completes with WFS_SUCCESS
11.		If there are additional bunches to deliver then this can be determined from the output parameter of the WFS_SRVE_CIM_ITEMSPRESENTED event or the WFS_INF_CIM_PRESENT_STATUS command. Tell the customer that the items were not accepted, and to take the items. The customer should be informed that the items will be returned in multiple bunches.	
12.	Customer takes the bunch of items.		WFS_SRVE_CIM_ITEMSTAKEN

13.		Close shutter.	WFS_CMD_CIM_CLOSE_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED) WFS_CMD_CIM_CLOSE_SHUTTER completes with WFS_SUCCESS
14.		If more refused items need to be taken: Repeat steps 9. – 14. Else Go to step 15.	
15.		Display the amount recognized so far.	
16.		Ask the customer for further actions: If the customer wants to deposit the amount: Continue with step 17. If the customer wants to get back all items inserted so far see table "Cancellation by Customer (Explicit Shutter Control)"	
17.		Transport the money into the cash units of type WFS_CIM_TYPERECYCLING / WFS_CIM_TYPECASHIN.	WFS_CMD_CIM_CASH_IN_END
18.		Credit the money to the customer's account.	
19.		End of transaction.	

8.19 Multiple Bunches Returned During WFS_CMD_CIM_CASH_IN_ROLLBACK (using WFS_CMD_CIM_PREPARE_PRESENT)

The following table describes the flow of a roll back operation where items are rolled back during the transaction. The application use WFS_CMD_CIM_PREPARE_PRESENT commands to move items to the output position. The Service Provider has explicit shutter control. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are used. Additionally, the number of items rolled back may be greater than the number of items that can be presented at the output position.

This flow covers the following cases:

- *bShutterControl* == FALSE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE, *bPresentControl* == FALSE, *bPreparePresent* == TRUE

Step	**Customer**	**Application**	**XFS Commands and Events**
1.-10.	See OK Transaction (Explicit Shutter Control).		
11.	Selection: Return all the items.		
12.		Transport the items recognized to the output position.	WFS_CMD_CIM_CASH_IN_ROLLBACK * WFS_CMD_CIM_CASH_IN_ROLLBACK reset the *lpTotalReturnedItems* output parameter of WFS_INF_CIM_PRESENT_STATUS.
			WFS_CMD_CIM_CASH_IN_ROLLBACK completes with WFS_SUCCESS.
13.		Move items to be rolled back to the output position.	WFS_CMD_CIM_PREPARE_PRESENT WFS_CMD_CIM_PREPARE_PRESENT completes with WFS_SUCCESS
14.		Open shutter.	WFS_CMD_CIM_OPEN_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED WFS_CMD_CIM_OPEN_SHUTTER completes with WFS_SUCCESS
15.		Tell the customer to take the items. The customer should be informed that the items will be returned in multiple bunches. If there are additional bunches to deliver then this can be determined from the output parameter of the WFS_SRVE_CIM_ITEMSPRESENTED event.	
16.	Customer takes the bunch of items.		WFS_SRVE_CIM_ITEMSTAKEN
17.		Close shutter.	WFS_CMD_CIM_CLOSE_SHUTTER … WFS_SRVE_CIM_SHUTTERSTATUS-CHANGED(WFS_CIM_SHTCLOSED) WFS_CMD_CIM_CLOSE_SHUTTER completes with WFS_SUCCESS

18.		If more items need to be taken:
		    Repeat steps 13. – 18.
		Else
		    Go to step 19.	

19.		End of transaction.	

9. ATM Mixed Media Transaction Flow – Application Guidelines

Compound CIM/IPM deposit devices are able to accept and process different types of media such as cash and checks. In order to improve the speed and usability of deposit devices it may be desirable to allow a bunch of items deposited to contain a variety of media types. Typically this is a bunch containing both cash and checks and is termed ‘Mixed Media processing’.

During this type of transaction the customer will insert cash and checks together in one bunch. The device will identify each item. Items not positively identified may be immediately returned to the customer. All remaining items can be deposited and shared deposit bins can be configured to receive mixed items. The application can also choose to return all items. Additionally the specification allows for depositing all checks and returning all cash or vice-versa depending on requirements.

In order to facilitate devices of differing hardware design and to support reuse of the XFS API, Mixed Media processing is achieved by initiating a CIM and an IPM transaction in parallel. The application and Service Providers must be able to handle concurrent CIM and IPM commands and events. The application will use the WFS_CMD_CIM_SET_MODE or WFS_CMD_IPM_SET_MODE command to activate Mixed Media processing. The literals used (i.e. WFS_CIM_IPMMIXEDMEDIA) describe the modes and indicate the nature of the compound device. This allows applications to open the correct interfaces to drive the transaction.

Mixed Media processing commands that move media in the device require commands to be called on both CIM and IPM interfaces. See the table below for a list of CIM commands and their IPM counterparts. Where the operation is to be cancelled the application is required to cancel only one command on either the CIM or IPM interface. Applications must be aware that the command that was NOT explicitly cancelled may complete with a WFS_ERR_CANCELED error.

For example the application must call both WFS_CMD_CIM_CASH_IN and WFS_CMD_IPM_MEDIA_IN commands to initiate the transaction. If an application wishes to cancel the transaction before items are inserted, only the WFS_CMD_CIM_CASH_IN command can be cancelled and the WFS_CMD_IPM_MEDIA_IN command will also be cancelled.

Devices suitable for Mixed Media processing must report WFSCIMCAPS.*bShutterControl* == TRUE to allow WFS_CMD_CIM_PRESENT_MEDIA and WFS_CMD_IPM_PRESENT_MEDIA commands to work concurrently.

The Mixed Media mode can be determined by calling WFS_INF_CIM_STATUS or WFS_INF_IPM_STATUS command and checking the value of the *wMixedMode* field.

Where an error occurs both CIM and IPM interfaces will report it. To recover the device a reset command can be called on either of the interfaces. Reset calls on both CIM and IPM interfaces are not required.

Application refusal (in the IPM interface) is not supported in Mixed Media mode.

To initiate a Mixed Media transaction the WFS_CMD_CIM_CASH_IN_START command must be called. There is no equivalent command to the WFS_CMD_CIM_CASH_IN_START command on the IPM interface.

Commands and their counterparts:

This table lists the counterpart IPM commands which must be called as well as the CIM commands when in Mixed Media processing mode.

CIM command	**IPM Command**
WFS_CMD_CIM_CASH_IN	WFS_CMD_IPM_MEDIA_IN
WFS_CMD_CIM_CASH_IN_END	WFS_CMD_IPM_MEDIA_IN_END or where *bMixedDepositAndRollback* is TRUE WFS_CMD_IPM_MEDIA_IN_ROLLBACK
WFS_CMD_CIM_CASH_IN_ROLLBACK	WFS_CMD_IPM_MEDIA_IN_ROLLBACK or where *bMixedDepositAndRollback* is TRUE WFS_CMD_IPM_MEDIA_IN_END
WFS_CMD_CIM_PRESENT_MEDIA	WFS_CMD_IPM_PRESENT_MEDIA
WFS_CMD_CIM_RETRACT	WFS_CMD_IPM_RETRACT_MEDIA

Events and their Counterparts

The CIM and IPM interfaces both have a range of events to inform the application of device activity. During Mixed Media processing events fired from each interface can describe the same situation (i.e. items presented). In these cases the recommendation to application developers is to rely on a single interface for these duplicate notifications. The choice of which interface to use to handle specific events will be based on factors such as current codebase or application presentation requirements.

CIM Event	**IPM Event**
WFS_USRE_CIM_CASHUNITTHRESHOLD	WFS_USRE_IPM_MEDIABINTHRESHOLD
WFS_SRVE_CIM_CASHUNITINFOCHANGED	WFS_SRVE_IPM_MEDIABININFOCHANGED
WFS_EXEE_CIM_CASHUNITERROR	WFS_EXEE_IPM_MEDIABINERROR
WFS_SRVE_CIM_ITEMSTAKEN	WFS_SRVE_IPM_MEDIATAKEN
WFS_SRVE_CIM_COUNTS_CHANGED	WFS_SRVE_IPM_MEDIABININFOCHANGED
WFS_EXEE_CIM_INPUTREFUSE	WFS_EXEE_IPM_MEDIAREFUSED
WFS_SRVE_CIM_ITEMSPRESENTED	WFS_EXEE_IPM_MEDIAPRESENTED
WFS_SRVE_CIM_ITEMSINSERTED	WFS_EXEE_IPM_MEDIAINSERTED
WFS_EXEE_CIM_SUBCASHIN	WFS_EXEE_IPM_MEDIADATA
WFS_SRVE_CIM_MEDIADETECTED	WFS_SRVE_IPM_MEDIADETECTED
WFS_EXEE_CIM_INSERTITEMS	WFS_EXEE_IPM_NOMEDIA
WFS_SRVE_CIM_DEVICEPOSITION	WFS_SRVE_IPM_DEVICEPOSITION
WFS_SRVE_CIM_POWER_SAVE_CHANGE	WFS_SRVE_IPM_POWER_SAVE_CHANGE

The following sections describe the flow of a Mixed Media transaction on a compound CIM/IPM device. These application flows are provided as guidelines only. In all cases WFSCIMPOSCAPS.*bPresentControl* == TRUE unless otherwise stated.

9.1 Mixed Media OK Transaction

The following table describes a normal Mixed Media transaction flow where there is a successful deposit.

This flow covers the following case:

- *bShutterControl* == TRUE, *wMixedMode* == WFS_CIM_IPMMIXEDMEDIA

Step	Application/Customer	CIM Commands and Events	IPM Commands and Events
1.	Application transaction opens sessions with both the CIM and the IPM service providers.		
2.	Customer selects Mixed Media transaction.	WFS_CMD_CIM_CASH_IN_START	
3.		* WFS_CMD_CIM_CASH_IN initiated (The shutter is not opened until WFS_CMD_IPM_MEDIA_IN is called.)	* WFS_CMD_IPM_MEDIA_IN initiated (Service Provider opens the input shutter).
4.		… WFS_SRVE_CIM_SHUTTER-STATUSCHANGED(WFS_CIM_SHT OPEN) WFS_EXEE_CIM_INSERTITEMS event is sent when the shutter is fully open and the device is ready to begin accepting items.	… WFS_SRVE_IPM_SHUTTER-STATUSCHANGED(WFS_IPM_SHT OPEN) WFS_EXEE_IPM_NOMEDIA This event specifies that media must be inserted into the device in order for the execute command to proceed.
5.	Ask the customer to insert items.		
6.	Customer inserts items.		
7.		WFS_SRVE_CIM_ITEMSINSERTED	WFS_EXEE_IPM_MEDIA-INSERTED
8.		The Service Provider closes the input shutter and the device begins processing the inserted items. … WFS_SRVE_CIM_SHUTTER-STATUSCHANGED(WFS_CIM_SHT CLOSED)	… WFS_SRVE_IPM_SHUTTER-STATUSCHANGED(WFS_IPM_SHT CLOSED) Send one WFS_EXEE_IPM_MEDIADATA event for every check item identified
9.		* WFS_CMD_CIM_CASH_IN completes.	* WFS_CMD_IPM_MEDIA_IN completes.
10.		WFS_INF_CIM_CASH_IN_STATUS can be issued to request the number of CIM related items that were inserted.	WFS_INF_IPM_TRANSACTION_-STATUS is issued to request the number of IPM related items that were inserted.
11.	Display the items recognized and associated information so far.		Process the checks by sending any of: WFS_CMD_IPM_READ_IMAGE, WFS_CMD_IPM_SET_-DESTINATION, WFS_CMD_IPM_PRINT_TEXT, WFS_CMD_IPM_GET_IMAGE_-AFTER_PRINT

12.	Ask the customer for further actions: If the customer wants to insert more items: Repeat from step 3. If the customer wants to finish the transaction: Continue with step 13. If the customer wants to get back all items inserted so far see table "Cancellation by Customer".		
13.		* WFS_CMD_CIM_CASH_IN_END initiated (The device will not complete the media movement until WFS_CMD_IPM_MEDIA_IN_END command is called on IPM interface.)	* WFS_CMD_IPM_MEDIA_IN_END initiated Print on individual media items (as specified from IPM commands)
14.		Transport the items into the specified destinations.	
15.		* WFS_CMD_CIM_CASH_IN_END completes.	* WFS_CMD_IPM_MEDIA_IN_END completes. Output parameter indicates media bin / outputs positions that have received items.
16.	Credit the appropriate funds to the customer's account.		
17.	End of transaction.		

9.2 Mixed Media Cancellation by Customer

The following table describes the flow of a Mixed Media transaction where the customer wants all the items to be returned. In this case the returned items must be explicitly presented by the application.

This flow covers the following cases:

- *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE
- *bCompound* == TRUE, *wMixedMode* == WFS_CIM_IPMMIXEDMEDIA
- WFSCIMPOSCAPS.*bPresentControl* == FALSE

Step	**Customer/ Application**	**CIM Commands and Events**	**IPM Commands and Events**
1.- 12.	As per OK Transaction.		
13.	Selection: Return all the items.		
14.	Transport the items recognized to the output position.	* WFS_CMD_CIM_CASH_IN_-ROLLBACK initiated (No physical action may take place until the WFS_CMD_IPM_-MEDIA_IN_ROLLBACK command.)	* WFS_CMD_IPM_MEDIA_IN_-ROLLBACK initiated
15.		* WFS_CMD_CIM_CASH_IN_-ROLLBACK completes.	* WFS_CMD_IPM_MEDIA_IN_-ROLLBACK completes.
16.		* WFS_CMD_CIM_PRESENT_MEDIA initiated (No physical action may take place until the WFS_CMD_IPM_PRESENT_MEDIA command.)	* WFS_CMD_IPM_PRESENT_MEDIA initiated
17.		The Service Provider opens the shutter(s). CIM cash moves to output position. … WFS_SRVE_CIM_SHUTTERSTATUS CHANGED(WFS_CIM_SHTOPEN)	The Service Provider opens the shutter(s). IPM media moves to output position. … WFS_SRVE_IPM_SHUTTERSTATUS CHANGED(WFS_IPM_SHTOPEN)
18.	Request removal of the items.	WFS_SRVE_CIM_ITEMSPRESENTED .	WFS_EXEE_IPM_MEDIA-PRESENTED.
19.		* WFS_CMD_CIM_PRESENT_MEDIA completes.	* WFS_CMD_IPM_PRESENT_MEDIA completes.
20.	Customer takes the items from the output position.		
21.		WFS_SRVE_CIM_ITEMSTAKEN	WFS_SRVE_IPM_MEDIATAKEN
22.		The Service Provider closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS CHANGED(WFS_CIM_SHTCLOSED)	… WFS_SRVE_IPM_SHUTTERSTATUS CHANGED(WFS_IPM_SHTCLOSED)
23.	End of transaction.		

9.3 Mixed Media Cancellation by Customer on Cash Part Only

The following table describes the flow of a Mixed Media transaction where the customer wants the cash items to be returned but deposit the check items. In this case the returned items are implicitly presented by the Service Provider.

This flow covers the following cases:

- *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE
- *wMixedMode* == WFS_CIM_IPMMIXEDMEDIA
- WFSCIMPOSCAPS.*bPresentControl* == TRUE

Step	**Customer/ Application**	**CIM Commands and Events**	**IPM Commands and Events**
1.- 12.	As per OK transaction		
13.	Selection: return cash items.		
14.	Transport the items recognized to the output position.	* WFS_CMD_CIM_CASH_IN_-ROLLBACK initiated (No physical action may take place until the WFS_CMD_IPM_MEDIA_IN_END command.)	* WFS_CMD_IPM_MEDIA_IN_END initiated
15.			Print on, and deposit individual media items (as specified by IPM commands).
16.		The Service Provider opens the shutter. CIM cash moves to output position. … WFS_SRVE_CIM_SHUTTERSTATUS CHANGED(WFS_CIM_SHTOPEN)	… WFS_SRVE_IPM_SHUTTERSTATUS CHANGED(WFS_IPM_SHTOPEN)
17.	Request removal of the cash items.	WFS_SRVE_CIM_ITEMSPRESENTED .	WFS_EXEE_IPM_MEDIA-PRESENTED.
18.		* WFS_CMD_CIM_CASH_IN_-ROLLBACK completes.	* WFS_CMD_IPM_MEDIA_IN_END completes.
19.	Customer takes the cash items from the output position.		
20.		WFS_SRVE_CIM_ITEMSTAKEN The Service Provider closes the shutter. … WFS_SRVE_CIM_SHUTTERSTATUS CHANGED(WFS_CIM_SHTCLOSED)	WFS_SRVE_IPM_MEDIATAKEN … WFS_SRVE_IPM_SHUTTERSTATUS CHANGED(WFS_IPM_SHTCLOSED)
21.	End of transaction.		

9.4 Mixed Media Multiple Refused Items

The following table describes the flow of a Mixed Media transaction where items are rejected during the transaction. Additionally, the number of items refused may be greater than the number of items that can be presented at the output position. In this case the returned items must be explicitly presented by the application.

This flow covers the following cases:

- *bShutterControl* == TRUE, *bItemsInsertedSensor* == TRUE, *bItemsTakenSensor* == TRUE
- *bCompound* == TRUE, *wMixedMode* == WFS_CIM_IPMMIXEDMEDIA
- WFSCIMPOSCAPS.*bPresentControl* == FALSE

Step	**Application/ Customer**	**CIM Commands and Events**	**IPM Commands and Events**
1.	Customer selects Mixed Media transaction.	WFS_CMD_CIM_CASH_IN_START	
2.		* WFS_CMD_CIM_CASH_IN initiated (The shutter is not opened until WFS_CMD_IPM_MEDIA_IN is called.) … WFS_SRVE_CIM_SHUTTERSTATUS CHANGED(WFS_CIM_SHTOPEN)	* WFS_CMD_IPM_MEDIA_IN initiated Service Provider opens the input shutter. … WFS_SRVE_CIM_SHUTTERSTATUS CHANGED(WFS_CIM_SHTOPEN)
3.		WFS_EXEE_CIM_INSERTITEMS event is sent when the shutter is fully open and the device is ready to begin accepting items.	WFS_EXEE_IPM_NOMEDIA This event specifies that media must be inserted into the device in order for the execute command to proceed.
4.	Ask the customer to insert items.		
5.	Customer inserts items.		
6.		WFS_SRVE_CIM_ITEMSINSERTED	WFS_EXEE_IPM_MEDIAINSERTED
7.		The Service Provider closes the input shutter and the device begins processing the inserted items. … WFS_SRVE_CIM_SHUTTERSTATUS CHANGED(WFS_CIM_SHTCLOSED)	… WFS_SRVE_IPM_SHUTTERSTATUS-CHANGED(WFS_IPM_SHTCLOSED) Send one WFS_EXEE_IPM_MEDIADATA event for every check item identified.
8.	Items are refused.	WFS_EXEE_CIM_INPUTREFUSE event sent with appropriate *lpusReason* parameter. Items that are not bills or checks are rejected with WFS_CIM_INVALIDBILL.	WFS_EXEE_IPM_MEDIAREFUSED
9.		* WFS_CMD_CIM_CASH_IN completes.	* WFS_CMD_IPM_MEDIA_IN completes.
10.	Application chooses to return refused items now.	* WFS_CMD_CIM_PRESENT_MEDIA initiated (No physical action may take place until the WFS_CMD_IPM_PRESENT_MEDIA command.)	* WFS_CMD_IPM_PRESENT_MEDIA initiated
11.	Each bunch of items presented.	… WFS_SRVE_CIM_SHUTTERSTATUS CHANGED(WFS_CIM_SHTOPEN) WFS_SRVE_CIM_ITEMSPRESENTED	… WFS_SRVE_IPM_SHUTTERSTATUS-CHANGED(WFS_IPM_SHTOPEN WFS_EXEE_IPM_MEDIAPRESENTED

12.	All but last bunch of items taken.	WFS_SRVE_CIM_ITEMSTAKEN	WFS_SRVE_IPM_MEDIATAKEN
13.		* WFS_CMD_CIM_PRESENT_MEDIA completes.	* WFS_CMD_IPM_PRESENT_MEDIA completes.
14.	Last bunch of items taken.	WFS_SRVE_CIM_ITEMSTAKEN	WFS_SRVE_IPM_MEDIATAKEN
15.	Transaction continues from step 13. in the OK transaction.		

10. Rules for Cash Unit Exchange

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

If a cash unit is removed from the CIM outside of the Start/End Exchange operations and subsequently reinserted the status of the physical cash unit should be set to WFS_CIM_STATCUMANIP to indicate to the application that the physical cash unit has been removed, reinserted and possibly tampered with. While the cash unit has this status the Service Provider should not attempt to use it as part of a cash-in operation. The WFS_CIM_STATCUMANIP status should not change until the next Start/End Exchange operation is performed, even if the cash unit is replaced in its original position.

If all the physical cash units belonging to a logical cash unit are manipulated the parent logical cash unit that the physical cash units belong to should also have its status set to WFS_CIM_STATCUMANIP.

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

Mixed Media Processing:

Where the device supports cash units that can store non-CIM items, a counters update to those cash units applied by the CIM interface can also be seen in the other interfaces available to the compound device.

The CIM *ulCount* on a shared bin (of type WFS_CIM_TYPECASHIN) reports the total number of banknotes, checks or coins of all types in the cash unit. This is for the following reasons:

1. *ulCount* on CIM has the same meaning as *ulCount* on IPM. That is the number of items of any type in the bin.

2. *ulMaximum*, is truly representative of the capacity of the physical bin and software thresholds can accurately reflect the state of the bin.

3. Use of *ulCount* representing items from both interfaces gives the greatest flexibility. Dedicated CIM or IPM bins and therefore counts can still be achieved through bin configuration.

4. The actual number of notes can be determined from *lpNoteNumberList.*

The following table describes the effect on the IPM counts where an application causes counter changes to a shared cassette using the CIM interface. The example assumes the starting position of a shared CIM cash unit/IPM media bin:

From WFSCIMCASHIN:
fwType = WFS_CIM_TYPECASHIN
fwItemType = WFS_CIM_CITYPALL|WFS_CIM_CITYPIPM
ulCashInCount = 0
ulCount = 0

And the IPM starting position for the shared CIM cash unit/IPM media bin:

From WFSIPMMEDIABIN:
fwType = WFS_IPM_TYPEMEDIAIN
wMediaType = WFS_IPM_MEDIATYPCOMPOUND
ulMediaInCount = 0
ulCount = 0

	Application Activity	CIM Counts on the shared cash unit	IPM Counts on the shared media bin
1.	A customer enters 10 good notes and 10 good checks in the same transaction.	*ulCashInCount* = 10 *ulCount* = 20	*ulMediaInCount* = 10 *ulCount* = 20
2.	Replenishment activity removes all items from the cash unit and clears the counts using WFS_CMD_CIM_END_EXCHANGE	*ulCashInCount* = 0 *ulCount* = 0	*ulMediaInCount* = 0 *ulCount* = 0

3.	A further customer enters 10 good notes and 10 good checks in the same transaction.	*ulCashInCount* = 10 *ulCount* = 20	*ulMediaInCount* = 10 *ulCount* = 20
4.	Replenishment activity removes only cash items from the cash unit. The CIM counts are adjusted using WFS_CMD_CIM_SET_CASH_UNIT_INFO *ulCashInCount* is set to 0, and *ulCount* is set to 10	*ulCashInCount* = 0 *ulCount* = 10	*ulMediaInCount* = 10 *ulCount* = 10
5.	A further customer enters 10 good notes and 10 good checks in the same transaction.	*ulCashInCount* = 10 *ulCount* = 30	*ulMediaInCount* = 20 *ulCount* = 30
6.	Replenishment activity removes only checks (20 items) from the cash unit. The counts are adjusted using WFS_CMD_IPM_SET_MEDIA_BIN_INFO. *ulMediaInCount* is set to 0, and *ulCount* is set to 10	*ulCashInCount* = 10 *ulCount* = 10	*ulMediaInCount* = 0 *ulCount* = 10

Multiple Physical Cash Units:

Where a logical cash unit contains more than one physical cash unit and is configured to accept or dispense more than one note ID, the breakdown of notes contained within each physical cash unit can be tracked or specified if the Service Provider supports the NOTENUMBERLIST string in the physical cash unit *lpszExtra* (see WFSCIMPHCU). Support for this is defined by the *bPhysicalNoteList* capability.

It is not mandatory to specify the NOTENUMBERLIST string in an Exchange even if supported; the Service Provider will track the counts from the point of the replenishment.

The following flow shows how this can be used:

	User/Application Activity	Logical Cash Unit Counts	Physical Cash Unit Counts
1	The device is replenished by inserting two physical cash units which are associated with one logical cash unit. The first physical cash unit contains 500 x usNoteID 1, the second cash unit contains 500 x usNoteID 2. Application performs an Exchange to set the counts including the NOTENUMBERLIST in the physical cash units.	*ulCount = 1000* *lpNoteNumberList:* *usNumOfNoteNumbers = 2* *lppNoteNumber[0].usNoteID = 1* *lppNoteNumber[0].ulCount = 500* *lppNoteNumber[1].usNoteID = 2* *lppNoteNumber[1].ulCount = 500*	lppPhysical[0]: *ulCount = 500* NOTENUMBERLIST=1,500 lppPhysical[1]: *ulCount = 500* NOTENUMBERLIST=2,500
2.	After several transactions, the first physical cash unit is full and requires replenishment. Application queries the counts.	*ulCount = 1600* *lpNoteNumberList:* *usNumOfNoteNumbers = 2* *lppNoteNumber[0].usNoteID = 1* *lppNoteNumber[0].ulCount = 900* *lppNoteNumber[1].usNoteID = 2* *lppNoteNumber[1].ulCount = 700*	lppPhysical[0]: *ulCount = 1000* NOTENUMBERLIST=1,800;2,200 lppPhysical[1]: *ulCount = 600* NOTENUMBERLIST=1,100;2,500

3.	The first physical cash unit is removed. The logical cash unit counts can now report only what is in the remaining physical cash unit. Application queries the counts	*ulCount* = 600 *lpNoteNumberList*: *usNumOfNoteNumbers* = 2 *lppNoteNumber[0].usNoteID* = 1 *lppNoteNumber[0].ulCount* = 100 *lppNoteNumber[1].usNoteID* = 2 *lppNoteNumber[1].ulCount* = 500	lppPhysical[0]: *ulCount* = 600 NOTENUMBERLIST=1,100;2,500
4.	A new cash unit is inserted containing 300 x usNoteID 1. As the application already knows the contents of the remaining physical cash unit, the logical cash unit counts can be calculated. Application performs an Exchange to set the counts.	*ulCount* = 900 *lpNoteNumberList*: *usNumOfNoteNumbers* = 2 *lppNoteNumber[0].usNoteID* = 1 *lppNoteNumber[0].ulCount* = 400 *lppNoteNumber[1].usNoteID* = 2 *lppNoteNumber[1].ulCount* = 500	lppPhysical[0]: *ulCount* = 300 NOTENUMBERLIST=1,300 lppPhysical[1]: *ulCount* = 600 NOTENUMBERLIST=1,100;2,500

11. Events Associated with Cash Unit Status Changes

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

11.1 One Physical Cash Unit Goes HIGH

The following table describes a deposit transaction case where the status of a physical cash unit only changes from WFS_CIM_STATCUOK to WFS_CIM_STATCUHIGH.

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

	Action	**Status/Event**
1.		Logical CU 1: WFS_CIM_STATCUOK - Physical CU 1: WFS_CIM_STATCUOK - Physical CU 2: WFS_CIM_STATCUOK
2.	A user deposits items.	
3.	The device accepts and moves the items into Physical CU 1, whose status changes to high.	
4.	The status of Logical CU 1 does not change.	Logical CU 1: WFS_CIM_STATCUOK - Physical CU 1: **WFS_CIM_STATCUHIGH** - Physical CU 2: WFS_CIM_STATCUOK WFS_SRVE_CIM_CASHUNITINFOCHANGED

11.2 Last Physical Cash Unit Goes HIGH

The following table describes a deposit transaction case where the status of a logical cash unit changes from WFS_CIM_STATCUOK to WFS_CIM_STATCUHIGH.

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

	Action	**Status/Event**
1.		Logical CU 1: WFS_CIM_STATCUOK - Physical CU 1: WFS_CIM_STATCUHIGH - Physical CU 2: WFS_CIM_STATCUOK
2.	A user deposits items.	
3.	The device accepts and moves the items into Physical CU 2, whose status changes to high.	
4.	As a result, the status of Logical CU 1 changes to high.	Logical CU 1: **WFS_CIM_STATCUHIGH** - Physical CU 1: WFS_CIM_STATCUHIGH - Physical CU 2: **WFS_CIM_STATCUHIGH** WFS_SRVE_CIM_CASHUNITINFOCHANGED WFS_USRE_CIM_CASHUNITTHRESHOLD

11.3 One Physical Cash Unit Goes INOP

The following table describes a deposit transaction case where the status of a logical cash unit changes from WFS_CIM_STATCUOK to WFS_CIM_STATCUHIGH as the result of a physical cash unit failure.

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

	Action	**Status/Event**
1.		Logical CU 1: WFS_CIM_STATCUOK - Physical CU 1: WFS_CIM_STATCUOK - Physical CU 2: WFS_CIM_STATCUHIGH
2.	A user deposits items.	
3.	The device accepts the items and tries to move them into Physical CU 1; however, a problem occurs in the cash unit, whose status changes to inoperative.	
4.	Instead, the device moves the items into Physical CU 2.	
5.	As a result, the status of Logical CU 1 changes to high.	Logical CU 1: **WFS_CIM_STATCUHIGH** - Physical CU 1: **WFS_CIM_STATCUINOP** - Physical CU 2: WFS_CIM_STATCUHIGH WFS_EXEE_CIM_CASHUNITERROR WFS_SRVE_CIM_CASHUNITINFOCHANGED WFS_USRE_CIM_CASHUNITTHRESHOLD

11.4 Last Physical Cash Unit Goes FULL

The following table describes a deposit transaction case where the status of a logical cash unit changes from WFS_CIM_STATCUHIGH to WFS_CIM_STATCUFULL.

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

	Action	**Status/Event**
1.		Logical CU 1: WFS_CIM_STATCUHIGH - Physical CU 1: WFS_CIM_STATCUFULL - Physical CU 2: WFS_CIM_STATCUHIGH
2.	A user deposits items.	
3.	The device accepts and moves the items into Physical CU 2, whose status changes to full.	
4.	As a result, the status of Logical CU 1 changes to full.	Logical CU 1: **WFS_CIM_STATCUFULL** - Physical CU 1: WFS_CIM_STATCUFULL - Physical CU 2: **WFS_CIM_STATCUFULL** WFS_SRVE_CIM_CASHUNITINFOCHANGED

12. C - Header file

```
/******************************************************************************
*                                                                             *
* xfscim.h      XFS - Cash Acceptor (CIM) definitions                         *
*                                                                             *
*               Version 3.~~30  (March 19 2015) ~~40  (December 6 2019)
*
*                                                                             *
******************************************************************************/

#ifndef __INC_XFSCIM__H
#define __INC_XFSCIM__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

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

/* values of WFSCIMCAPS.wClass */

#define     WFS_SERVICE_CLASS_CIM              (13)
#define     WFS_SERVICE_CLASS_VERSION_CIM      (~~0x1E03~~0x2803) /* Version 3.~~30~~40 */
#define     WFS_SERVICE_CLASS_NAME_CIM         "CIM"

#define     CIM_SERVICE_OFFSET                 (WFS_SERVICE_CLASS_CIM * 100)

/* CIM Info Commands */

#define     WFS_INF_CIM_STATUS                 (CIM_SERVICE_OFFSET + 1)
#define     WFS_INF_CIM_CAPABILITIES           (CIM_SERVICE_OFFSET + 2)
#define     WFS_INF_CIM_CASH_UNIT_INFO         (CIM_SERVICE_OFFSET + 3)
#define     WFS_INF_CIM_TELLER_INFO            (CIM_SERVICE_OFFSET + 4)
#define     WFS_INF_CIM_CURRENCY_EXP           (CIM_SERVICE_OFFSET + 5)
#define     WFS_INF_CIM_BANKNOTE_TYPES         (CIM_SERVICE_OFFSET + 6)
#define     WFS_INF_CIM_CASH_IN_STATUS         (CIM_SERVICE_OFFSET + 7)
#define     WFS_INF_CIM_GET_P6_INFO            (CIM_SERVICE_OFFSET + 8)
#define     WFS_INF_CIM_GET_P6_SIGNATURE       (CIM_SERVICE_OFFSET + 9)
#define     WFS_INF_CIM_GET_ITEM_INFO          (CIM_SERVICE_OFFSET + 10)
#define     WFS_INF_CIM_POSITION_CAPABILITIES  (CIM_SERVICE_OFFSET + 11)
#define     WFS_INF_CIM_REPLENISH_TARGET       (CIM_SERVICE_OFFSET + 12)
#define     WFS_INF_CIM_DEVICELOCK_STATUS      (CIM_SERVICE_OFFSET + 13)
#define     WFS_INF_CIM_CASH_UNIT_CAPABILITIES (CIM_SERVICE_OFFSET + 14)
#define     WFS_INF_CIM_DEPLETE_SOURCE         (CIM_SERVICE_OFFSET + 15)
#define     WFS_INF_CIM_GET_ALL_ITEMS_INFO     (CIM_SERVICE_OFFSET + 16)
#define     WFS_INF_CIM_GET_BLACKLIST          (CIM_SERVICE_OFFSET + 17)
#define     WFS_INF_CIM_GET_CLASSIFICATION_LIST (CIM_SERVICE_OFFSET + 18)
#define     WFS_INF_CIM_CASH_UNIT_COUNT_STATUS (CIM_SERVICE_OFFSET + 19)
#define     WFS_INF_CIM_PRESENT_STATUS         (CIM_SERVICE_OFFSET + 20)


/* CIM Execute Commands */

#define     WFS_CMD_CIM_CASH_IN_START          (CIM_SERVICE_OFFSET + 1)
#define     WFS_CMD_CIM_CASH_IN                (CIM_SERVICE_OFFSET + 2)
#define     WFS_CMD_CIM_CASH_IN_END            (CIM_SERVICE_OFFSET + 3)
#define     WFS_CMD_CIM_CASH_IN_ROLLBACK       (CIM_SERVICE_OFFSET + 4)
#define     WFS_CMD_CIM_RETRACT                (CIM_SERVICE_OFFSET + 5)
#define     WFS_CMD_CIM_OPEN_SHUTTER           (CIM_SERVICE_OFFSET + 6)
#define     WFS_CMD_CIM_CLOSE_SHUTTER          (CIM_SERVICE_OFFSET + 7)
#define     WFS_CMD_CIM_SET_TELLER_INFO        (CIM_SERVICE_OFFSET + 8)
#define     WFS_CMD_CIM_SET_CASH_UNIT_INFO     (CIM_SERVICE_OFFSET + 9)
#define     WFS_CMD_CIM_START_EXCHANGE         (CIM_SERVICE_OFFSET + 10)
#define     WFS_CMD_CIM_END_EXCHANGE           (CIM_SERVICE_OFFSET + 11)
#define     WFS_CMD_CIM_OPEN_SAFE_DOOR         (CIM_SERVICE_OFFSET + 12)
```

```
#define     WFS_CMD_CIM_RESET                    (CIM_SERVICE_OFFSET + 13)
#define     WFS_CMD_CIM_CONFIGURE_CASH_IN_UNITS  (CIM_SERVICE_OFFSET + 14)
#define     WFS_CMD_CIM_CONFIGURE_NOTETYPES      (CIM_SERVICE_OFFSET + 15)
#define     WFS_CMD_CIM_CREATE_P6_SIGNATURE      (CIM_SERVICE_OFFSET + 16)
#define     WFS_CMD_CIM_SET_GUIDANCE_LIGHT       (CIM_SERVICE_OFFSET + 17)
#define     WFS_CMD_CIM_CONFIGURE_NOTE_READER    (CIM_SERVICE_OFFSET + 18)
#define     WFS_CMD_CIM_COMPARE_P6_SIGNATURE     (CIM_SERVICE_OFFSET + 19)
#define     WFS_CMD_CIM_POWER_SAVE_CONTROL       (CIM_SERVICE_OFFSET + 20)
#define     WFS_CMD_CIM_REPLENISH                (CIM_SERVICE_OFFSET + 21)
#define     WFS_CMD_CIM_SET_CASH_IN_LIMIT        (CIM_SERVICE_OFFSET + 22)
#define     WFS_CMD_CIM_CASH_UNIT_COUNT          (CIM_SERVICE_OFFSET + 23)
#define     WFS_CMD_CIM_DEVICE_LOCK_CONTROL      (CIM_SERVICE_OFFSET + 24)
#define     WFS_CMD_CIM_SET_MODE                 (CIM_SERVICE_OFFSET + 25)
#define     WFS_CMD_CIM_PRESENT_MEDIA            (CIM_SERVICE_OFFSET + 26)
#define     WFS_CMD_CIM_DEPLETE                  (CIM_SERVICE_OFFSET + 27)
#define     WFS_CMD_CIM_SET_BLACKLIST            (CIM_SERVICE_OFFSET + 28)
#define     WFS_CMD_CIM_SYNCHRONIZE_COMMAND      (CIM_SERVICE_OFFSET + 29)
#define     WFS_CMD_CIM_SET_CLASSIFICATION_LIST  (CIM_SERVICE_OFFSET + 30)
#define     WFS_CMD_CIM_PREPARE_PRESENT          (CIM_SERVICE_OFFSET + 31)


/* CIM Messages */

#define     WFS_SRVE_CIM_SAFEDOOROPEN            (CIM_SERVICE_OFFSET + 1)
#define     WFS_SRVE_CIM_SAFEDOORCLOSED          (CIM_SERVICE_OFFSET + 2)
#define     WFS_USRE_CIM_CASHUNITTHRESHOLD       (CIM_SERVICE_OFFSET + 3)
#define     WFS_SRVE_CIM_CASHUNITINFOCHANGED     (CIM_SERVICE_OFFSET + 4)
#define     WFS_SRVE_CIM_TELLERINFOCHANGED       (CIM_SERVICE_OFFSET + 5)
#define     WFS_EXEE_CIM_CASHUNITERROR           (CIM_SERVICE_OFFSET + 6)
#define     WFS_SRVE_CIM_ITEMSTAKEN              (CIM_SERVICE_OFFSET + 7)
#define     WFS_SRVE_CIM_COUNTS_CHANGED          (CIM_SERVICE_OFFSET + 8)
#define     WFS_EXEE_CIM_INPUTREFUSE             (CIM_SERVICE_OFFSET + 9)
#define     WFS_SRVE_CIM_ITEMSPRESENTED          (CIM_SERVICE_OFFSET + 10)
#define     WFS_SRVE_CIM_ITEMSINSERTED           (CIM_SERVICE_OFFSET + 11)
#define     WFS_EXEE_CIM_NOTEERROR               (CIM_SERVICE_OFFSET + 12)
#define     WFS_EXEE_CIM_SUBCASHIN               (CIM_SERVICE_OFFSET + 13)
#define     WFS_SRVE_CIM_MEDIADETECTED           (CIM_SERVICE_OFFSET + 14)
#define     WFS_EXEE_CIM_INPUT_P6                (CIM_SERVICE_OFFSET + 15)
#define     WFS_EXEE_CIM_INFO_AVAILABLE          (CIM_SERVICE_OFFSET + 16)
#define     WFS_EXEE_CIM_INSERTITEMS             (CIM_SERVICE_OFFSET + 17)
#define     WFS_SRVE_CIM_DEVICEPOSITION          (CIM_SERVICE_OFFSET + 18)
#define     WFS_SRVE_CIM_POWER_SAVE_CHANGE       (CIM_SERVICE_OFFSET + 19)
#define     WFS_EXEE_CIM_INCOMPLETEREPLENISH     (CIM_SERVICE_OFFSET + 20)
#define     WFS_EXEE_CIM_INCOMPLETEDEPLETE       (CIM_SERVICE_OFFSET + 21)
#define     WFS_SRVE_CIM_SHUTTERSTATUSCHANGED    (CIM_SERVICE_OFFSET + 22)
#define     WFS_SRVE_CIM_COUNTACCURACYCHANGED    (CIM_SERVICE_OFFSET + 23)

/* values of WFSCIMSTATUS.fwDevice */

#define     WFS_CIM_DEVONLINE                   WFS_STAT_DEVONLINE
#define     WFS_CIM_DEVOFFLINE                  WFS_STAT_DEVOFFLINE
#define     WFS_CIM_DEVPOWEROFF                 WFS_STAT_DEVPOWEROFF
#define     WFS_CIM_DEVNODEVICE                 WFS_STAT_DEVNODEVICE
#define     WFS_CIM_DEVUSERERROR                WFS_STAT_DEVUSERERROR
#define     WFS_CIM_DEVHWERROR                  WFS_STAT_DEVHWERROR
#define     WFS_CIM_DEVBUSY                     WFS_STAT_DEVBUSY
#define     WFS_CIM_DEVFRAUDATTEMPT             WFS_STAT_DEVFRAUDATTEMPT
#define     WFS_CIM_DEVPOTENTIALFRAUD           WFS_STAT_DEVPOTENTIALFRAUD

/* values of WFSCIMSTATUS.fwSafeDoor */

#define     WFS_CIM_DOORNOTSUPPORTED             (1)
#define     WFS_CIM_DOOROPEN                     (2)
#define     WFS_CIM_DOORCLOSED                   (3)
#define     WFS_CIM_DOORUNKNOWN                  (4)

/* values of WFSCIMSTATUS.fwAcceptor */

#define     WFS_CIM_ACCOK                        (0)
#define     WFS_CIM_ACCCUSTATE                   (1)
```

```
#define     WFS_CIM_ACCCUSTOP                   (2)
#define     WFS_CIM_ACCCUUNKNOWN                (3)

/* values of WFSCIMSTATUS.fwIntermediateStacker */

#define     WFS_CIM_ISEMPTY                     (0)
#define     WFS_CIM_ISNOTEMPTY                  (1)
#define     WFS_CIM_ISFULL                      (2)
#define     WFS_CIM_ISUNKNOWN                   (4)
#define     WFS_CIM_ISNOTSUPPORTED              (5)

/* Size and max index of dwGuidLights array */
#define     WFS_CIM_GUIDLIGHTS_SIZE             (32)
#define     WFS_CIM_GUIDLIGHTS_MAX              (WFS_CIM_GUIDLIGHTS_SIZE - 1)

/* Indices of WFSCIMSTATUS.dwGuidLights [...]
              WFSCIMCAPS.dwGuidLights [...]
*/

#define     WFS_CIM_GUIDANCE_POSINNULL          (0)
#define     WFS_CIM_GUIDANCE_POSINLEFT          (1)
#define     WFS_CIM_GUIDANCE_POSINRIGHT         (2)
#define     WFS_CIM_GUIDANCE_POSINCENTER        (3)
#define     WFS_CIM_GUIDANCE_POSINTOP           (4)
#define     WFS_CIM_GUIDANCE_POSINBOTTOM        (5)
#define     WFS_CIM_GUIDANCE_POSINFRONT         (6)
#define     WFS_CIM_GUIDANCE_POSINREAR          (7)
#define     WFS_CIM_GUIDANCE_POSOUTLEFT         (8)
#define     WFS_CIM_GUIDANCE_POSOUTRIGHT        (9)
#define     WFS_CIM_GUIDANCE_POSOUTCENTER       (10)
#define     WFS_CIM_GUIDANCE_POSOUTTOP          (11)
#define     WFS_CIM_GUIDANCE_POSOUTBOTTOM       (12)
#define     WFS_CIM_GUIDANCE_POSOUTFRONT        (13)
#define     WFS_CIM_GUIDANCE_POSOUTREAR         (14)
#define     WFS_CIM_GUIDANCE_POSOUTNULL         (15)

/* Values of WFSCIMSTATUS.dwGuidLights [...]
             WFSCIMCAPS.dwGuidLights [...]
*/

#define     WFS_CIM_GUIDANCE_NOT_AVAILABLE      (0x00000000)
#define     WFS_CIM_GUIDANCE_OFF                (0x00000001)
#define     WFS_CIM_GUIDANCE_SLOW_FLASH         (0x00000004)
#define     WFS_CIM_GUIDANCE_MEDIUM_FLASH       (0x00000008)
#define     WFS_CIM_GUIDANCE_QUICK_FLASH        (0x00000010)
#define     WFS_CIM_GUIDANCE_CONTINUOUS         (0x00000080)
#define     WFS_CIM_GUIDANCE_RED                (0x00000100)
#define     WFS_CIM_GUIDANCE_GREEN              (0x00000200)
#define     WFS_CIM_GUIDANCE_YELLOW             (0x00000400)
#define     WFS_CIM_GUIDANCE_BLUE               (0x00000800)
#define     WFS_CIM_GUIDANCE_CYAN               (0x00001000)
#define     WFS_CIM_GUIDANCE_MAGENTA            (0x00002000)
#define     WFS_CIM_GUIDANCE_WHITE              (0x00004000)
#define     WFS_CIM_GUIDANCE_ENTRY              (0x00100000)
#define     WFS_CIM_GUIDANCE_EXIT               (0x00200000)

/* values of WFSCIMSTATUS.wDevicePosition
             WFSCIMDEVICEPOSITION.wPosition */

#define     WFS_CIM_DEVICEINPOSITION            (0)
#define     WFS_CIM_DEVICENOTINPOSITION         (1)
#define     WFS_CIM_DEVICEPOSUNKNOWN            (2)
#define     WFS_CIM_DEVICEPOSNOTSUPP            (3)

/* values of WFSCIMSTATUS.fwStackerItems */

#define     WFS_CIM_CUSTOMERACCESS              (0)
#define     WFS_CIM_NOCUSTOMERACCESS            (1)
#define     WFS_CIM_ACCESSUNKNOWN               (2)
#define     WFS_CIM_NOITEMS                     (4)
```

```
/* values of WFSCIMSTATUS.fwBankNoteReader */

#define     WFS_CIM_BNROK                       (0)
#define     WFS_CIM_BNRINOP                     (1)
#define     WFS_CIM_BNRUNKNOWN                  (2)
#define     WFS_CIM_BNRNOTSUPPORTED             (3)

/* values of WFSCIMSTATUS.fwShutter */

#define     WFS_CIM_SHTCLOSED                   (0)
#define     WFS_CIM_SHTOPEN                     (1)
#define     WFS_CIM_SHTJAMMED                   (2)
#define     WFS_CIM_SHTUNKNOWN                  (3)
#define     WFS_CIM_SHTNOTSUPPORTED             (4)

/* values of WFSCIMCAPS.wMixedMode */

#define     WFS_CIM_MIXEDMEDIANOTSUPP           (0)
#define     WFS_CIM_IPMMIXEDMEDIA               (1)

/* values of WFSCIMSETMODE.wMixedMode */
/* values of WFSCIMSTATUS.wMixedMode.*/

#define     WFS_CIM_MIXEDMEDIANOTACTIVE         (0)

/* values of WFSCIMINPOS.fwPositionStatus */

#define     WFS_CIM_PSEMPTY                     (0)
#define     WFS_CIM_PSNOTEMPTY                  (1)
#define     WFS_CIM_PSUNKNOWN                   (2)
#define     WFS_CIM_PSNOTSUPPORTED              (3)
#define     WFS_CIM_PSFOREIGNITEMS              (4)

/* values of WFSCIMSTATUS.fwTransport */

#define     WFS_CIM_TPOK                        (0)
#define     WFS_CIM_TPINOP                      (1)
#define     WFS_CIM_TPUNKNOWN                   (2)
#define     WFS_CIM_TPNOTSUPPORTED              (3)

/* values of WFSCIMINPOS.fwTransportStatus */

#define     WFS_CIM_TPSTATEMPTY                 (0)
#define     WFS_CIM_TPSTATNOTEMPTY              (1)
#define     WFS_CIM_TPSTATNOTEMPTYCUST          (2)
#define     WFS_CIM_TPSTATNOTEMPTY_UNK          (3)
#define     WFS_CIM_TPSTATNOTSUPPORTED          (4)

/* values of WFSCIMOUTPOS.fwJammedShutterPosition */

#define     WFS_CIM_SHUTTERPOS_NOTSUPPORTED     (0)
#define     WFS_CIM_SHUTTERPOS_NOTJAMMED        (1)
#define     WFS_CIM_SHUTTERPOS_OPEN             (2)
#define     WFS_CIM_SHUTTERPOS_PARTIALLY_OPEN   (3)
#define     WFS_CIM_SHUTTERPOS_CLOSED           (4)
#define     WFS_CIM_SHUTTERPOS_UNKNOWN          (5)

/* values of WFSCIMCAPS.fwType */

#define     WFS_CIM_TELLERBILL                  (0)
#define     WFS_CIM_SELFSERVICEBILL             (1)
#define     WFS_CIM_TELLERCOIN                  (2)
#define     WFS_CIM_SELFSERVICECOIN             (3)

/* values of WFSCIMCAPS.fwExchangeType */
/* values of WFSCIMSTARTEX.fwExchangeType */

#define     WFS_CIM_EXBYHAND                    (0x0001)
#define     WFS_CIM_EXTOCASSETTES               (0x0002)
```

```
#define     WFS_CIM_CLEARRECYCLER                 (0x0004)
#define     WFS_CIM_DEPOSITINTO                   (0x0008)

/* values of WFSCIMCAPS.fwRetractTransportActions */
/* values of WFSCIMCAPS.fwRetractStackerActions */

#define     WFS_CIM_PRESENT                       (0x0001)
#define     WFS_CIM_RETRACT                       (0x0002)
#define     WFS_CIM_NOTSUPP                       (0x0004)
#define     WFS_CIM_REJECT                        (0x0008)
#define     WFS_CIM_BILLCASSETTES                 (0x0010)
#define     WFS_CIM_CASHIN                        (0x0020)

/* values for WFSCIMCAPS.fwCashInLimit */

#define     WFS_CIM_LIMITNOTSUPP                  (0x0000)
#define     WFS_CIM_LIMITBYTOTALITEMS             (0x0001)
#define     WFS_CIM_LIMITBYAMOUNT                 (0x0002)
#define     WFS_CIM_LIMITMULTIPLE                 (0x0004)
#define     WFS_CIM_LIMITREFUSEOTHER              (0x0008)

/* values of WFSCIMCASHIN.fwType */

#define     WFS_CIM_TYPERECYCLING                 (1)
#define     WFS_CIM_TYPECASHIN                    (2)
#define     WFS_CIM_TYPEREPCONTAINER              (3)
#define     WFS_CIM_TYPERETRACTCASSETTE           (4)
#define     WFS_CIM_TYPEREJECT                    (5)
#define     WFS_CIM_TYPECDMSPECIFIC               (6)

/* values of WFSCIMCASHIN.fwItemType */
/* values of WFSCIMCASHINTYPE.dwType */

#define     WFS_CIM_CITYPALL                      (0x0001)
#define     WFS_CIM_CITYPUNFIT                    (0x0002)
#define     WFS_CIM_CITYPINDIVIDUAL               (0x0004)
#define     WFS_CIM_CITYPLEVEL3                   (0x0008)
#define     WFS_CIM_CITYPLEVEL2                   (0x0010)
#define     WFS_CIM_CITYPIPM                      (0x0020)
#define     WFS_CIM_CITYPLEVEL1                   (0x0040)
#define     WFS_CIM_CITYPUNFITINDIVIDUAL          (0x0080)

/* values of WFSCIMCASHIN.usStatus */
/* values of WFSCIMPHCU.usPStatus */

#define     WFS_CIM_STATCUOK                      (0)
#define     WFS_CIM_STATCUFULL                    (1)
#define     WFS_CIM_STATCUHIGH                    (2)
#define     WFS_CIM_STATCULOW                     (3)
#define     WFS_CIM_STATCUEMPTY                   (4)
#define     WFS_CIM_STATCUINOP                    (5)
#define     WFS_CIM_STATCUMISSING                 (6)
#define     WFS_CIM_STATCUNOVAL                   (7)
#define     WFS_CIM_STATCUNOREF                   (8)  /* NOTE: Not used in CIM */
#define     WFS_CIM_STATCUMANIP                   (9)

/* values of WFSCIMSTATUS.fwPositions */
/* values of WFSCIMCAPS.fwPositions */
/* values of WFSCIMINPOS.fwPosition */
/* values of WFSCIMTELLERDETAILS.fwInputPosition */
/* values of WFSCIMCASHINSTART.fwInputPosition */
/* values of WFSCIMMOVEITEMS.fwPosition */

#define     WFS_CIM_POSNULL                       (0x0000)
#define     WFS_CIM_POSINLEFT                     (0x0001)
#define     WFS_CIM_POSINRIGHT                    (0x0002)
#define     WFS_CIM_POSINCENTER                   (0x0004)
#define     WFS_CIM_POSINTOP                      (0x0008)
#define     WFS_CIM_POSINBOTTOM                   (0x0010)
#define     WFS_CIM_POSINFRONT                    (0x0020)
```

```
#define     WFS_CIM_POSINREAR                   (0x0040)

/* values of WFSCIMSTATUS.fwPositions */
/* values of WFSCIMCAPS.fwPositions */
/* values of WFSCIMTELLERDETAILS.fwOutputPosition */
/* values of WFSCIMCASHINSTART.fwOutputPosition */
/* values of WFSCIMOUTPUT.fwPosition */
/* values of WFSCIMMOVEITEMS.fwPosition */

#define     WFS_CIM_POSOUTLEFT                  (0x0080)
#define     WFS_CIM_POSOUTRIGHT                 (0x0100)
#define     WFS_CIM_POSOUTCENTER                (0x0200)
#define     WFS_CIM_POSOUTTOP                   (0x0400)
#define     WFS_CIM_POSOUTBOTTOM                (0x0800)
#define     WFS_CIM_POSOUTFRONT                 (0x1000)
#define     WFS_CIM_POSOUTREAR                  (0x2000)

/* values of WFSCIMCASHINSTATUS.wStatus */

#define     WFS_CIM_CIOK                        (0)
#define     WFS_CIM_CIROLLBACK                  (1)
#define     WFS_CIM_CIACTIVE                    (2)
#define     WFS_CIM_CIRETRACT                   (3)
#define     WFS_CIM_CIUNKNOWN                   (4)
#define     WFS_CIM_CIRESET                     (5)

/* values of WFSCIMCAPS.fwRetractAreas */
/* values of WFSCIMRETRACT.usRetractArea */

#define     WFS_CIM_RA_RETRACT                  (0x0001)
#define     WFS_CIM_RA_TRANSPORT                (0x0002)
#define     WFS_CIM_RA_STACKER                  (0x0004)
#define     WFS_CIM_RA_BILLCASSETTES            (0x0008)
#define     WFS_CIM_RA_NOTSUPP                  (0x0010)
#define     WFS_CIM_RA_REJECT                   (0x0020)
#define     WFS_CIM_RA_CASHIN                   (0x0040)

/* values of WFSCIMP6INFO.usLevel */
/* values of WFSCIMP6SIGNATURE.usLevel */
/* values of WFSCIMGETALLITEMSINFO.usLevel */
/* values of WFSCIMITEMINFOALL.usLevel */

#define     WFS_CIM_LEVEL_1                     (1)
#define     WFS_CIM_LEVEL_2                     (2)
#define     WFS_CIM_LEVEL_3                     (3)
#define     WFS_CIM_LEVEL_4                     (4)

/* values of WFSCIMITEMINFOALL.usLevel */

#define     WFS_CIM_LEVEL_ALL                   (0)

/* values of WFSCIMTELLERUPDATE.usAction */

#define     WFS_CIM_CREATE_TELLER               (1)
#define     WFS_CIM_MODIFY_TELLER               (2)
#define     WFS_CIM_DELETE_TELLER               (3)

/* values of WFSCIMCUERROR.wFailure */

#define     WFS_CIM_CASHUNITEMPTY               (1)
#define     WFS_CIM_CASHUNITERROR               (2)
#define     WFS_CIM_CASHUNITFULL                (3)
#define     WFS_CIM_CASHUNITLOCKED              (4)
#define     WFS_CIM_CASHUNITNOTCONF             (5)
#define     WFS_CIM_CASHUNITINVALID             (6)
#define     WFS_CIM_CASHUNITCONFIG              (7)
#define     WFS_CIM_FEEDMODULEPROBLEM           (8)
#define     WFS_CIM_CASHUNITPHYSICALLOCKED      (9)
#define     WFS_CIM_CASHUNITPHYSICALUNLOCKED    (10)
```

```
/*values of WFSCIMP6SIGNATURE.dwOrientation*/

#define     WFS_CIM_ORFRONTTOP                  (1)
#define     WFS_CIM_ORFRONTBOTTOM               (2)
#define     WFS_CIM_ORBACKTOP                   (3)
#define     WFS_CIM_ORBACKBOTTOM                (4)
#define     WFS_CIM_ORUNKNOWN                   (5)
#define     WFS_CIM_ORNOTSUPPORTED              (6)

/* values for WFSCIMGETITEMINFO.dwItemInfoType */
#define     WFS_CIM_ITEM_NOTSUPP                (0x00000000)
#define     WFS_CIM_ITEM_SERIALNUMBER           (0x00000001)
#define     WFS_CIM_ITEM_SIGNATURE              (0x00000002)
#define     WFS_CIM_ITEM_IMAGEFILE              (0x00000004)

/* values of lpusReason in WFS_EXEE_CIM_INPUTREFUSE */

#define     WFS_CIM_CASHINUNITFULL              (1)
#define     WFS_CIM_INVALIDBILL                 (2)
#define     WFS_CIM_NOBILLSTODEPOSIT            (3)
#define     WFS_CIM_DEPOSITFAILURE              (4)
#define     WFS_CIM_COMMINPCOMPFAILURE          (5)
#define     WFS_CIM_STACKERFULL                 (6)
#define     WFS_CIM_FOREIGN_ITEMS_DETECTED      (7)
#define     WFS_CIM_INVALIDBUNCH                (8)
#define     WFS_CIM_COUNTERFEIT                 (9)
#define     WFS_CIM_LIMITOVERTOTALITEMS         (10)
#define     WFS_CIM_LIMITOVERAMOUNT             (11)

/* values of lpusReason in WFS_EXEE_CIM_NOTESERROR */

#define     WFS_CIM_DOUBLENOTEDETECTED          (1)
#define     WFS_CIM_LONGNOTEDETECTED            (2)
#define     WFS_CIM_SKEWEDNOTE                  (3)
#define     WFS_CIM_INCORRECTCOUNT              (4)
#define     WFS_CIM_NOTESTOOCLOSE               (5)
#define     WFS_CIM_OTHERNOTEERROR              (6)
#define     WFS_CIM_SHORTNOTEDETECTED           (7)

/* Values of fwUsage in WFS_INF_CIM_POSITION_CAPABILITIES */

#define     WFS_CIM_POSIN                       (0x0001)
#define     WFS_CIM_POSREFUSE                   (0x0002)
#define     WFS_CIM_POSROLLBACK                 (0x0004)

/* values of WFSCIMPOSITIONINFO.wAdditionalBunches */
/* values of WFSCIMPRESENTSTATUS.wAdditionalBunches */

#define     WFS_CIM_ADDBUNCHNONE                (1)
#define     WFS_CIM_ADDBUNCHONEMORE             (2)
#define     WFS_CIM_ADDBUNCHUNKNOWN             (3)

/* values of WFSCIMPOSITIONINFO.usBunchesRemaining */
/* values of WFSCIMPRESENTSTATUS.usBunchesRemaining */

#define     WFS_CIM_NUMBERUNKNOWN               (255)

/* values of WFSCIMCAPS.fwCountActions */

#define     WFS_CIM_COUNTNOTSUPP                (0x0000)
#define     WFS_CIM_COUNTINDIVIDUAL             (0x0001)
#define     WFS_CIM_COUNTALL                    (0x0002)

/* values of WFSCIMDEVICELOCKCONTROL.wDeviceAction */
/* values of WFSCIMDEVICELOCKCONTROL.wCashUnitAction */
/* values of WFSCIMUNITLOCKCONTROL.wUnitAction */

#define     WFS_CIM_LOCK                        (1)
#define     WFS_CIM_UNLOCK                      (2)
#define     WFS_CIM_LOCKALL                     (3)
```

```
#define     WFS_CIM_UNLOCKALL                   (4)
#define     WFS_CIM_LOCKINDIVIDUAL              (5)
#define     WFS_CIM_NOLOCKACTION                (6)
#define     WFS_CIM_LOCKUNKNOWN                 (7)
#define     WFS_CIM_LOCKNOTSUPPORTED            (8)

/* values of WFSCIMSTATUS.wAntiFraudModule */

#define     WFS_CIM_AFMNOTSUPP                  (0)
#define     WFS_CIM_AFMOK                       (1)
#define     WFS_CIM_AFMINOP                     (2)
#define     WFS_CIM_AFMDEVICEDETECTED           (3)
#define     WFS_CIM_AFMUNKNOWN                  (4)

/* values for WFSCIMITEMINFOALL.wOnBlacklist */

#define     WFS_CIM_ONBLACKLIST                 (0x0001)
#define     WFS_CIM_NOTONBLACKLIST              (0x0002)
#define     WFS_CIM_BLACKLISTUNKNOWN            (0x0003)

/* values for WFSCIMITEMINFOALL.wItemLocation */

#define     WFS_CIM_LOCATION_DEVICE             (0x0001)
#define     WFS_CIM_LOCATION_CASHUNIT           (0x0002)
#define     WFS_CIM_LOCATION_CUSTOMER           (0x0003)
#define     WFS_CIM_LOCATION_UNKNOWN            (0x0004)

/* values for WFSCIMITEMINFOALL.wOnClassificationList */

#define     WFS_CIM_CLASSIFICATIONLIST_ON       (0x0001)
#define     WFS_CIM_CLASSIFICATIONLIST_NOTON    (0x0002)
#define     WFS_CIM_CLASSIFICATIONLIST_UNKNOWN  (0x0003)

/* values for WFSCIMCASHUNITCOUNTSTATUS.usAccuracy */
/* values for WFSCIMPHCUCOUNTSTATUS.usAccuracy */

#define     WFS_CIM_ACCURACYNOTSUPPORTED        (0)
#define     WFS_CIM_COUNTACCURATE               (1)
#define     WFS_CIM_COUNTACCURATESET            (2)
#define     WFS_CIM_COUNTINACCURATE             (3)
#define     WFS_CIM_ACCURACYUNKNOWN             (4)

/* values for WFSCIMITEMINFOALL.wItemDeviceLocation */

#define     WFS_CIM_DEVLOC_STACKER              (0x0001)
#define     WFS_CIM_DEVLOC_OUTPUT               (0x0002)
#define     WFS_CIM_DEVLOC_TRANSPORT            (0x0003)
#define     WFS_CIM_DEVLOC_UNKNOWN              (0x0004)

/* values of WFSCIMPRESENTSTATUS.wPresentState */
#define     WFS_CIM_PRESENTED                   (1)
#define     WFS_CIM_NOTPRESENTED                (2)
#define     WFS_CIM_UNKNOWN                     (3)

/* XFS CIM Errors */

#define WFS_ERR_CIM_INVALIDCURRENCY             (-(CIM_SERVICE_OFFSET + 0))
#define WFS_ERR_CIM_INVALIDTELLERID             (-(CIM_SERVICE_OFFSET + 1))
#define WFS_ERR_CIM_CASHUNITERROR               (-(CIM_SERVICE_OFFSET + 2))
#define WFS_ERR_CIM_TOOMANYITEMS                (-(CIM_SERVICE_OFFSET + 7))
#define WFS_ERR_CIM_UNSUPPOSITION               (-(CIM_SERVICE_OFFSET + 8))
#define WFS_ERR_CIM_SAFEDOOROPEN                (-(CIM_SERVICE_OFFSET + 10))
#define WFS_ERR_CIM_SHUTTERNOTOPEN              (-(CIM_SERVICE_OFFSET + 12))
#define WFS_ERR_CIM_SHUTTEROPEN                 (-(CIM_SERVICE_OFFSET + 13))
#define WFS_ERR_CIM_SHUTTERCLOSED               (-(CIM_SERVICE_OFFSET + 14))
#define WFS_ERR_CIM_INVALIDCASHUNIT             (-(CIM_SERVICE_OFFSET + 15))
#define WFS_ERR_CIM_NOITEMS                     (-(CIM_SERVICE_OFFSET + 16))
#define WFS_ERR_CIM_EXCHANGEACTIVE              (-(CIM_SERVICE_OFFSET + 17))
#define WFS_ERR_CIM_NOEXCHANGEACTIVE            (-(CIM_SERVICE_OFFSET + 18))
#define WFS_ERR_CIM_SHUTTERNOTCLOSED            (-(CIM_SERVICE_OFFSET + 19))
```

```
#define WFS_ERR_CIM_ITEMSTAKEN                  (-(CIM_SERVICE_OFFSET + 23))
#define WFS_ERR_CIM_CASHINACTIVE                (-(CIM_SERVICE_OFFSET + 25))
#define WFS_ERR_CIM_NOCASHINACTIVE              (-(CIM_SERVICE_OFFSET + 26))
#define WFS_ERR_CIM_POSITION_NOT_EMPTY          (-(CIM_SERVICE_OFFSET + 28))
#define WFS_ERR_CIM_INVALIDRETRACTPOSITION      (-(CIM_SERVICE_OFFSET + 34))
#define WFS_ERR_CIM_NOTRETRACTAREA              (-(CIM_SERVICE_OFFSET + 35))
#define WFS_ERR_CIM_INVALID_PORT                (-(CIM_SERVICE_OFFSET + 36))
#define WFS_ERR_CIM_FOREIGN_ITEMS_DETECTED      (-(CIM_SERVICE_OFFSET + 37))
#define WFS_ERR_CIM_LOADFAILED                  (-(CIM_SERVICE_OFFSET + 38))
#define WFS_ERR_CIM_CASHUNITNOTEMPTY            (-(CIM_SERVICE_OFFSET + 39))
#define WFS_ERR_CIM_INVALIDREFSIG               (-(CIM_SERVICE_OFFSET + 40))
#define WFS_ERR_CIM_INVALIDTRNSIG               (-(CIM_SERVICE_OFFSET + 41))
#define WFS_ERR_CIM_POWERSAVETOOSHORT           (-(CIM_SERVICE_OFFSET + 42))
#define WFS_ERR_CIM_POWERSAVEMEDIAPRESENT       (-(CIM_SERVICE_OFFSET + 43))
#define WFS_ERR_CIM_DEVICELOCKFAILURE           (-(CIM_SERVICE_OFFSET + 44))
#define WFS_ERR_CIM_TOOMANYITEMSTOCOUNT         (-(CIM_SERVICE_OFFSET + 45))
#define WFS_ERR_CIM_COUNTPOSNOTEMPTY            (-(CIM_SERVICE_OFFSET + 46))
#define WFS_ERR_CIM_MEDIAINACTIVE               (-(CIM_SERVICE_OFFSET + 47))
#define WFS_ERR_CIM_COMMANDUNSUPP               (-(CIM_SERVICE_OFFSET + 48))
#define WFS_ERR_CIM_SYNCHRONIZEUNSUPP           (-(CIM_SERVICE_OFFSET + 49))

/*=================================================================*/
/* CIM Info Command Structures */
/*=================================================================*/

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

typedef struct _wfs_cim_status
{
    WORD                        fwDevice;
    WORD                        fwSafeDoor;
    WORD                        fwAcceptor;
    WORD                        fwIntermediateStacker;
    WORD                        fwStackerItems;
    WORD                        fwBanknoteReader;
    BOOL                        bDropBox;
    LPWFSCIMINPOS               *lppPositions;
    LPSTR                       lpszExtra;
    DWORD                       dwGuidLights[WFS_CIM_GUIDLIGHTS_SIZE];
    WORD                        wDevicePosition;
    USHORT                      usPowerSaveRecoveryTime;
    WORD                        wMixedMode;
    WORD                        wAntiFraudModule;
} WFSCIMSTATUS, *LPWFSCIMSTATUS;

typedef struct _wfs_cim_caps
{
    WORD                        wClass;
    WORD                        fwType;
    WORD                        wMaxCashInItems;
    BOOL                        bCompound;
    BOOL                        bShutter;
    BOOL                        bShutterControl;
    BOOL                        bSafeDoor;
    BOOL                        bCashBox;
    BOOL                        bRefill;
    WORD                        fwIntermediateStacker;
    BOOL                        bItemsTakenSensor;
    BOOL                        bItemsInsertedSensor;
    WORD                        fwPositions;
    WORD                        fwExchangeType;
    WORD                        fwRetractAreas;
    WORD                     fwRetractTransportActions;
    WORD                     fwRetractStackerActions;
    LPSTR                    lpszExtra;
    DWORD                    dwGuidLights[WFS_CIM_GUIDLIGHTS_SIZE];
    DWORD                    dwItemInfoTypes;
    BOOL                     bCompareSignatures;
    BOOL                     bPowerSaveControl;
    BOOL                     bReplenish;
    WORD                     fwCashInLimit;
    WORD                     fwCountActions;
    BOOL                     bDeviceLockControl;
    WORD                     wMixedMode;
    BOOL                     bMixedDepositAndRollback;
    BOOL                     bAntiFraudModule;
    BOOL                     bDeplete;
    BOOL                     bBlacklist;
    LPDWORD                  lpdwSynchronizableCommands;
    BOOL                     bClassificationList;
    BOOL                     bPhysicalNoteList;
} WFSCIMCAPS, *LPWFSCIMCAPS;

typedef struct _wfs_cim_physicalcu
{
    LPSTR                    lpPhysicalPositionName;
    CHAR                     cUnitID[5];
    ULONG                    ulCashInCount;
    ULONG                    ulCount;
    ULONG                    ulMaximum;
    USHORT                   usPStatus;
    BOOL                     bHardwareSensors;
    LPSTR                    lpszExtra;
    ULONG                    ulInitialCount;
    ULONG                    ulDispensedCount;
    ULONG                    ulPresentedCount;
    ULONG                    ulRetractedCount;
    ULONG                    ulRejectCount;
} WFSCIMPHCU, *LPWFSCIMPHCU;

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

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

typedef struct _wfs_cim_cash_in
{
    USHORT                   usNumber;
    DWORD                    fwType;
    DWORD                    fwItemType;
    CHAR                     cUnitID[5];
    CHAR                     cCurrencyID[3];
    ULONG                    ulValues;
    ULONG                    ulCashInCount;
    ULONG                    ulCount;
    ULONG                    ulMaximum;
    USHORT                   usStatus;
    BOOL                     bAppLock;
    LPWFSCIMNOTENUMBERLIST   lpNoteNumberList;
    USHORT                   usNumPhysicalCUs;
    LPWFSCIMPHCU             *lppPhysical;
    LPSTR                    lpszExtra;
    LPUSHORT                 lpusNoteIDs;
    WORD                     usCDMType;
    LPSTR                    lpszCashUnitName;
    ULONG                        ulInitialCount;
    ULONG                        ulDispensedCount;
    ULONG                        ulPresentedCount;
    ULONG                        ulRetractedCount;
    ULONG                        ulRejectCount;
    ULONG                        ulMinimum;
} WFSCIMCASHIN, *LPWFSCIMCASHIN;

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

typedef struct _wfs_cim_teller_info
{
    USHORT                       usTellerID;
    CHAR                         cCurrencyID[3];
} WFSCIMTELLERINFO, *LPWFSCIMTELLERINFO;

typedef struct _wfs_cim_teller_totals
{
   CHAR                          cCurrencyID[3];
   ULONG                         ulItemsReceived;
   ULONG                         ulItemsDispensed;
   ULONG                         ulCoinsReceived;
   ULONG                         ulCoinsDispensed;
   ULONG                         ulCashBoxReceived;
   ULONG                         ulCashBoxDispensed;
} WFSCIMTELLERTOTALS, *LPWFSCIMTELLERTOTALS;

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

typedef struct _wfs_cim_currency_exp
{
    CHAR                         cCurrencyID[3];
    SHORT                        sExponent;
} WFSCIMCURRENCYEXP, *LPWFSCIMCURRENCYEXP;


typedef struct _wfs_cim_note_type
{
    USHORT                       usNoteID;
    CHAR                         cCurrencyID[3];
    ULONG                        ulValues;
    USHORT                       usRelease;
    BOOL                         bConfigured;
} WFSCIMNOTETYPE, *LPWFSCIMNOTETYPE;

typedef struct _wfs_cim_note_type_list
{
    USHORT                       usNumOfNoteTypes;
    LPWFSCIMNOTETYPE             *lppNoteTypes;
} WFSCIMNOTETYPELIST, *LPWFSCIMNOTETYPELIST;

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

```
typedef struct _wfs_cim_P6_info
{
    USHORT                    usLevel;
    LPWFSCIMNOTENUMBERLIST    lpNoteNumberList;
    USHORT                    usNumOfSignatures;
} WFSCIMP6INFO, *LPWFSCIMP6INFO;

typedef struct _wfs_cim_get_P6_signature
{
    USHORT                    usLevel;
    USHORT                    usIndex;
} WFSCIMGETP6SIGNATURE, *LPWFSCIMGETP6SIGNATURE;

typedef struct _wfs_cim_P6_signature
{
    USHORT                    usNoteId;
    ULONG                     ulLength;
    DWORD                     dwOrientation;
    LPVOID                    lpSignature;
} WFSCIMP6SIGNATURE, *LPWFSCIMP6SIGNATURE;

typedef struct _wfs_cim_get_item_info
{
    USHORT                    usLevel;
    USHORT                    usIndex;
    DWORD                     dwItemInfoType;
} WFSCIMGETITEMINFO, *LPWFSCIMGETITEMINFO;

typedef struct _wfs_cim_get_all_items_info
{
    USHORT                    usLevel;
} WFSCIMGETALLITEMSINFO, *LPWFSCIMGETALLITEMSINFO;

typedef struct _wfs_cim_item_info_all
{
    USHORT                    usLevel;
    USHORT                    usNoteID;
    LPWSTR                    lpszSerialNumber;
    DWORD                     dwOrientation;
    LPSTR                     lpszP6SignatureFileName;
    LPSTR                     lpszImageFileName;
    WORD                      wOnBlacklist;
    WORD                      wItemLocation;
    USHORT                    usNumber;
    WORD                      wOnClassificationList;
    WORD                      wItemDeviceLocation;
} WFSCIMITEMINFOALL, *LPWFSCIMITEMINFOALL;

typedef struct _wfs_cim_all_items_info
{
    USHORT                    usCount;
    LPWFSCIMITEMINFOALL       *lppItemsList;
} WFSCIMALLITEMSINFO, *LPWFSCIMALLITEMSINFO;

typedef struct _wfs_cim_item_info
{
    USHORT                    usNoteID;
    LPWSTR                    lpszSerialNumber;
    LPWFSCIMP6SIGNATURE       lpP6Signature;
    LPSTR                     lpszImageFileName;
} WFSCIMITEMINFO, *LPWFSCIMITEMINFO;

typedef struct _wfs_cim_item_info_summary
{
    USHORT                    usLevel;
    USHORT                    usNumOfItems;
} WFSCIMITEMINFOSUMMARY, *LPWFSCIMITEMINFOSUMMARY;

typedef struct _wfs_cim_pos_caps
{
    WORD                        fwPosition;
    WORD                        fwUsage;
    BOOL                        bShutterControl;
    BOOL                        bItemsTakenSensor;
    BOOL                        bItemsInsertedSensor;
    WORD                        fwRetractAreas;
    LPSTR                       lpszExtra;
    BOOL                        bPresentControl;
    BOOL                        bPreparePresent;
} WFSCIMPOSCAPS, *LPWFSCIMPOSCAPS;

typedef struct _wfs_cim_pos_capabilities
{
    LPWFSCIMPOSCAPS             *lppPosCapabilities;
} WFSCIMPOSCAPABILITIES, *LPWFSCIMPOSCAPABILITIES;

typedef struct _wfs_cim_replenish_info
{
    USHORT                      usNumberSource;
} WFSCIMREPINFO, *LPWFSCIMREPINFO;

typedef struct _wfs_cim_replenish_info_target
{
    USHORT                      usNumberTarget;
} WFSCIMREPINFOTARGET, *LPWFSCIMREPINFOTARGET;

typedef struct _wfs_cim_replenish_info_result
{
    LPWFSCIMREPINFOTARGET       *lppReplenishTargets;
} WFSCIMREPINFORES, *LPWFSCIMREPINFORES;

typedef struct _wfs_cim_cash_unit_lock
{
    LPSTR                       lpPhysicalPositionName;
    WORD                        wCashUnitLockStatus;
} WFSCIMCASHUNITLOCK, *LPWFSCIMCASHUNITLOCK;

typedef struct _wfs_cim_device_lock_status
{
    WORD                        wDeviceLockStatus;
    LPWFSCIMCASHUNITLOCK        *lppCashUnitLock;
} WFSCIMDEVICELOCKSTATUS, *LPWFSCIMDEVICELOCKSTATUS;

typedef struct _wfs_cim_physicalcu_capabilities
{
    LPSTR                        lpPhysicalPositionName;
    ULONG                        ulMaximum;
    BOOL                         bHardwareSensors;
    LPSTR                        lpszExtra;
} WFSCIMPHCUCAPABILITIES, *LPWFSCIMPHCUCAPABILITIES;

typedef struct _wfs_cim_cash_unit_capabilities
{
    USHORT                      usNumber;
    USHORT                      usNumPhysicalCUs;
    LPWFSCIMPHCUCAPABILITIES    *lppPhysical;
    BOOL                        bRetractNoteCountThresholds;
    LPSTR                       lpszExtra;
    DWORD                       fwPossibleItemTypes;
} WFSCIMCASHUNITCAPABILITIES, *LPWFSCIMCASHUNITCAPABILITIES;

typedef struct _wfs_cim_cash_caps
{
    USHORT                          usCount;
    LPWFSCIMCASHUNITCAPABILITIES *lppCashUnitCaps;
} WFSCIMCASHCAPABILITIES, *LPWFSCIMCASHCAPABILITIES;

typedef struct _wfs_cim_deplete_info
{
    USHORT                      usNumberTarget;
```

```
} WFSCIMDEPINFO, *LPWFSCIMDEPINFO;

typedef struct _wfs_cim_deplete_info_source
{
    USHORT                      usNumberSource;
} WFSCIMDEPINFOSOURCE, *LPWFSCIMDEPINFOSOURCE;

typedef struct _wfs_cim_deplete_info_result
{
    LPWFSCIMDEPINFOSOURCE      *lppDepleteSources;
} WFSCIMDEPINFORES, *LPWFSCIMDEPINFORES;

typedef struct _wfs_cim_phcu_count_status
{
    LPSTR                       lpPhysicalPositionName;
    USHORT                      usAccuracy;
    LPSTR                       lpszExtra;
} WFSCIMPHCUCOUNTSTATUS, *LPWFSCIMPHCUCOUNTSTATUS;

typedef struct _wfs_cim_cash_unit_count_status
{
    USHORT                      usNumber;
    USHORT                      usAccuracy;
    USHORT                      usNumPhysicalCUs;
    LPWFSCIMPHCUCOUNTSTATUS    *lppPhCashUnitStatus;
    LPSTR                       lpszExtra;
} WFSCIMCASHUNITCOUNTSTATUS, *LPWFSCIMCASHUNITCOUNTSTATUS;

typedef struct _wfs_cim_cash_count_status
{
    USHORT                         usCount;
    LPWFSCIMCASHUNITCOUNTSTATUS *lppCashUnitStatus;
} WFSCIMCASHCOUNTSTATUS, *LPWFSCIMCASHCOUNTSTATUS;

typedef struct _wfs_cim_present_status
{
    WORD                        fwPosition;
    WORD                        wPresentState;
    WORD                        wAdditionalBunches;
    USHORT                      usBunchesRemaining;
    LPWFSCIMNOTENUMBERLIST      lpReturnedItems;
    LPWFSCIMNOTENUMBERLIST      lpTotalReturnedItems;
    LPWFSCIMNOTENUMBERLIST      lpRemainingItems;
    LPSTR                       lpszExtra;
} WFSCIMPRESENTSTATUS, *LPWFSCIMPRESENTSTATUS;

/*=================================================================*/
/* CIM Execute Command Structures */
/*=================================================================*/

typedef struct _wfs_cim_cash_in_start
{
    USHORT                      usTellerID;
    BOOL                        bUseRecycleUnits;
    WORD                        fwOutputPosition;
    WORD                        fwInputPosition;
} WFSCIMCASHINSTART, *LPWFSCIMCASHINSTART;

typedef struct _wfs_cim_retract
{
    WORD                        fwOutputPosition;
    USHORT                      usRetractArea;
    USHORT                      usIndex;
} WFSCIMRETRACT, *LPWFSCIMRETRACT;

typedef struct _wfs_cim_teller_update
{
    USHORT                      usAction;
    LPWFSCIMTELLERDETAILS       lpTellerDetails;
} WFSCIMTELLERUPDATE, *LPWFSCIMTELLERUPDATE;
```

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

typedef struct _wfs_cim_start_ex
{
    WORD                      fwExchangeType;
    USHORT                    usTellerID;
    USHORT                    usCount;
    LPUSHORT                  lpusCUNumList;
    LPWFSCIMOUTPUT            lpOutput;
} WFSCIMSTARTEX, *LPWFSCIMSTARTEX;

typedef struct _wfs_cim_itemposition
{
    USHORT                    usNumber;
    LPWFSCIMRETRACT           lpRetractArea;
    WORD                      fwOutputPosition;
} WFSCIMITEMPOSITION, *LPWFSCIMITEMPOSITION;

typedef struct _wfs_cim_cash_in_type
{
    USHORT                    usNumber;
    DWORD                     dwType;
    LPUSHORT                  lpusNoteIDs;
} WFSCIMCASHINTYPE, *LPWFSCIMCASHINTYPE;

typedef struct _wfs_cim_set_guidlight
{
    WORD                      wGuidLight;
    DWORD                     dwCommand;
} WFSCIMSETGUIDLIGHT, *LPWFSCIMSETGUIDLIGHT;

typedef struct _wfs_cim_configure_note_reader
{
    BOOL                      bLoadAlways;
} WFSCIMCONFIGURENOTEREADER, *LPWFSCIMCONFIGURENOTEREADER;

typedef struct _wfs_cim_configure_note_reader_out
{
    BOOL                      bRebootNecessary;
} WFSCIMCONFIGURENOTEREADEROUT, *LPWFSCIMCONFIGURENOTEREADEROUT;

typedef struct _wfs_cim_P6_compare_signature
{
   LPWFSCIMP6SIGNATURE       *lppP6ReferenceSignatures;
   LPWFSCIMP6SIGNATURE       *lppP6Signatures;
} WFSCIMP6COMPARESIGNATURE, *LPWFSCIMP6COMPARESIGNATURE;

typedef struct _wfs_cim_P6_signatures_index
{
    USHORT                    usIndex;
    USHORT                    usConfidenceLevel;
    ULONG                     ulLength;
    LPVOID                    lpComparisonData;
} WFSCIMP6SIGNATURESINDEX, *LPWFSCIMP6SIGNATURESINDEX;

typedef struct _wfs_cim_P6_compare_result
{
   USHORT                    usCount;
   LPWFSCIMP6SIGNATURESINDEX *lppP6SignaturesIndex;
} WFSCIMP6COMPARERESULT, *LPWFSCIMP6COMPARERESULT;

typedef struct _wfs_cim_power_save_control
{
    USHORT                    usMaxPowerSaveRecoveryTime;
} WFSCIMPOWERSAVECONTROL, *LPWFSCIMPOWERSAVECONTROL;

typedef struct _wfs_cim_replenish_target
{
    USHORT                    usNumberTarget;
    ULONG                     ulNumberOfItemsToMove;
    BOOL                      bRemoveAll;
} WFSCIMREPTARGET, *LPWFSCIMREPTARGET;

typedef struct _wfs_cim_replenish
{
    USHORT                    usNumberSource;
    LPWFSCIMREPTARGET         *lppReplenishTargets;
} WFSCIMREP, *LPWFSCIMREP;

typedef struct _wfs_cim_replenish_target_result
{
    USHORT                    usNumberTarget;
    USHORT                    usNoteID;
    ULONG                     ulNumberOfItemsReceived;
} WFSCIMREPTARGETRES, *LPWFSCIMREPTARGETRES;

typedef struct _wfs_cim_replenish_result
{
    ULONG                     ulNumberOfItemsRemoved;
    ULONG                     ulNumberOfItemsRejected;
    LPWFSCIMREPTARGETRES      *lppReplenishTargetResults;
} WFSCIMREPRES, *LPWFSCIMREPRES;

typedef struct _wfs_cim_amount_limit
{
    CHAR                      cCurrencyID[3];
    ULONG                     ulAmount;
} WFSCIMAMOUNTLIMIT, *LPWFSCIMAMOUNTLIMIT;

typedef struct _wfs_cim_cash_in_limit
{
    ULONG                     ulTotalItemsLimit;
    LPWFSCIMAMOUNTLIMIT       lpAmountLimit;
} WFSCIMCASHINLIMIT, *LPWFSCIMCASHINLIMIT;

typedef struct _wfs_cim_count
{
    USHORT                    usCount;
    LPUSHORT                  lpusCUNumList;
} WFSCIMCOUNT, *LPWFSCIMCOUNT;

typedef struct _wfs_cim_unit_lock_control
{
    LPSTR                     lpPhysicalPositionName;
    WORD                      wUnitAction;
} WFSCIMUNITLOCKCONTROL, *LPWFSCIMUNITLOCKCONTROL;

typedef struct _wfs_cim_device_lock_control
{
    WORD                      wDeviceAction;
    WORD                      wCashUnitAction;
    LPWFSCIMUNITLOCKCONTROL   *lppUnitLockControl;
} WFSCIMDEVICELOCKCONTROL, *LPWFSCIMDEVICELOCKCONTROL;

typedef struct _wfs_cim_setmode
{
    WORD                      wMixedMode;
} WFSCIMSETMODE, *LPWFSCIMSETMODE;

typedef struct _wfs_cim_present
{
    WORD                      fwPosition;
} WFSCIMPRESENT, *LPWFSCIMPRESENT;
```

```
typedef struct _wfs_cim_deplete_source
{
    USHORT                    usNumberSource;
    ULONG                     ulNumberOfItemsToMove;
    BOOL                      bRemoveAll;
} WFSCIMDEPSOURCE, *LPWFSCIMDEPSOURCE;

typedef struct _wfs_cim_deplete
{
    LPWFSCIMDEPSOURCE         *lppDepleteSources;
    USHORT                    usNumberTarget;
} WFSCIMDEP, *LPWFSCIMDEP;

typedef struct _wfs_cim_deplete_source_result
{
    USHORT                    usNumberSource;
    USHORT                    usNoteID;
    ULONG                     ulNumberOfItemsRemoved;
} WFSCIMDEPSOURCERES, *LPWFSCIMDEPSOURCERES;

typedef struct _wfs_cim_deplete_result
{
    ULONG                     ulNumberOfItemsReceived;
    ULONG                     ulNumberOfItemsRejected;
    LPWFSCIMDEPSOURCERES      *lppDepleteSourceResults;
} WFSCIMDEPRES, *LPWFSCIMDEPRES;

typedef struct _wfs_cim_blacklist_element
{
    LPWSTR                    lpszSerialNumber;
    CHAR                      cCurrencyID[3];
    ULONG                     ulValue;
} WFSCIMBLACKLISTELEMENT, *LPWFSCIMBLACKLISTELEMENT;

typedef struct _wfs_cim_blacklist
{
    LPWSTR                    lpszVersion;
    USHORT                    usCount;
    LPWFSCIMBLACKLISTELEMENT *lppBlacklistElements;
} WFSCIMBLACKLIST, *LPWFSCIMBLACKLIST;

typedef struct _wfs_cim_synchronize_command
{
    DWORD                     dwCommand;
    LPVOID                    lpCmdData;
} WFSCIMSYNCHRONIZECOMMAND, *LPWFSCIMSYNCHRONIZECOMMAND;

typedef struct _wfs_cim_classification_element
{
    LPWSTR                    lpszSerialNumber;
    CHAR                      cCurrencyID[3];
    ULONG                     ulValue;
    USHORT                    usLevel;
    BOOL                      bUnfit;
} WFSCIMCLASSIFICATIONELEMENT, *LPWFSCIMCLASSIFICATIONELEMENT;

typedef struct _wfs_cim_classification_list
{
    LPWSTR                    lpszVersion;
    USHORT                    usCount;
    LPWFSCIMCLASSIFICATIONELEMENT *lppClassificationElements;
} WFSCIMCLASSIFICATIONLIST, *LPWFSCIMCLASSIFICATIONLIST;

typedef struct _wfs_cim_moveitems
{
    WORD                      fwPosition;
} WFSCIMMOVEITEMS, *LPWFSCIMMOVEITEMS;

/*=================================================================*/
/* CIM Message Structures */
```

```
/*=================================================================*/

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

typedef struct _wfs_cim_counts_changed
{
    USHORT                  usCount;
    LPUSHORT                lpusCUNumList;
} WFSCIMCOUNTSCHANGED, *LPWFSCIMCOUNTSCHANGED;

typedef struct _wfs_cim_position_info
{
    WORD                    wPosition;
    WORD                    wAdditionalBunches;
    USHORT                  usBunchesRemaining;
} WFSCIMPOSITIONINFO, *LPWFSCIMPOSITIONINFO;

typedef struct _wfs_cim_device_position
{
    WORD                    wPosition;
} WFSCIMDEVICEPOSITION, *LPWFSCIMDEVICEPOSITION;

typedef struct _wfs_cim_power_save_change
{
    USHORT                  usPowerSaveRecoveryTime;
} WFSCIMPOWERSAVECHANGE, *LPWFSCIMPOWERSAVECHANGE;

typedef struct _wfs_cim_incomplete_replenish
{
    LPWFSCIMREPRES          lpReplenish;
} WFSCIMINCOMPLETEREPLENISH, *LPWFSCIMINCOMPLETEREPLENISH;

typedef struct _wfs_cim_incomplete_deplete
{
    LPWFSCIMDEPRES          lpDeplete;
} WFSCIMINCOMPLETEDEPLETE, *LPWFSCIMINCOMPLETEDEPLETE;

typedef struct _wfs_cim_shutter_status_changed
{
    WORD                    fwPosition;
    WORD                    fwShutter;
} WFSCIMSHUTTERSTATUSCHANGED, *LPWFSCIMSHUTTERSTATUSCHANGED;

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

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

#endif  /* __INC_XFSCIM__H */
```