CEN

WORKSHOP

AGREEMENT

CWA 15748-15

July 2008

ICS 35.240.50

English version

Extensions for Financial Services (XFS) interface specification - Release 3.10 - Part 15: Cash-In Module Device Class Interface - Programmer's Reference

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

Management Centre: rue de Stassart, 36 B-1050 Brussels

Ref. No.:CWA 15748-15:2008 D/E/F

Table of Contents

Foreword 4

1. Introduction 7
   1.1 Background to Release 3.10 7
   1.2 XFS Service-Specific Programming 7

2. Cash-In Module 8

3. References 9

4. Info Commands 10
   4.1 WFS_INF_CIM_STATUS 10
   4.2 WFS_INF_CIM_CAPABILITIES 15
   4.3 WFS_INF_CIM_CASH_UNIT_INFO 19
   4.4 WFS_INF_CIM_TELLER_INFO 27
   4.5 WFS_INF_CIM_CURRENCY_EXP 29
   4.6 WFS_INF_CIM_BANKNOTE_TYPES 30
   4.7 WFS_INF_CIM_CASH_IN_STATUS 31
   4.8 WFS_INF_CIM_GET_P6_INFO 32
   4.9 WFS_INF_CIM_GET_P6_SIGNATURE 33
   4.10 WFS_INF_CIM_GET_ITEM_INFO 35
   4.11 WFS_INF_CIM_POSITION_CAPABILITIES 37

5. Execute Commands 39
   5.1 WFS_CMD_CIM_CASH_IN_START 39
   5.2 WFS_CMD_CIM_CASH_IN 41
   5.3 WFS_CMD_CIM_CASH_IN_END 43
   5.4 WFS_CMD_CIM_CASH_IN_ROLLBACK 44
   5.5 WFS_CMD_CIM_RETRACT 46
   5.6 WFS_CMD_CIM_OPEN_SHUTTER 48
   5.7 WFS_CMD_CIM_CLOSE_SHUTTER 49
   5.8 WFS_CMD_CIM_SET_TELLER_INFO 50
   5.9 WFS_CMD_CIM_SET_CASH_UNIT_INFO 51
   5.10 WFS_CMD_CIM_START_EXCHANGE 53
   5.11 WFS_CMD_CIM_END_EXCHANGE 56
   5.12 WFS_CMD_CIM_OPEN_SAFE_DOOR 57
   5.13 WFS_CMD_CIM_RESET 58
   5.14 WFS_CMD_CIM_CONFIGURE_CASH_IN_UNITS 60
   5.15 WFS_CMD_CIM_CONFIGURE_NOTETYPES 61
   5.16 WFS_CMD_CIM_CREATE_P6_SIGNATURE 62
   5.17 WFS_CMD_CIM_SET_GUIDANCE_LIGHT 64

5.18 WFS_CMD_CIM_CONFIGURE_NOTE_READER 65
5.19 WFS_CMD_CIM_COMPARE_P6_SIGNATURE 66
5.20 WFS_CMD_CIM_POWER_SAVE_CONTROL 68

6. Events 69
6.1 WFS_SRVE_CIM_SAFEDOOROPEN 69
6.2 WFS_SRVE_CIM_SAFEDOORCLOSED 70
6.3 WFS_USRE_CIM_CASHUNITTHRESHOLD 71
6.4 WFS_SRVE_CIM_CASHUNITINFOCHANGED 72
6.5 WFS_SRVE_CIM_TELLERINFOCHANGED 73
6.6 WFS_EXEE_CIM_CASHUNITERROR 74
6.7 WFS_SRVE_CIM_ITEMSTAKEN 75
6.8 WFS_SRVE_CIM_COUNTS_CHANGED 76
6.9 WFS_EXEE_CIM_INPUTREFUSE 77
6.10 WFS_SRVE_CIM_ITEMSPRESENTED 78
6.11 WFS_SRVE_CIM_ITEMSINSERTED 79
6.12 WFS_EXEE_CIM_NOTEERROR 80
6.13 WFS_EXEE_CIM_SUBCASHIN 81
6.14 WFS_SRVE_CIM_MEDIADETECTED 82
6.15 WFS_EXEE_CIM_INPUT_P6 83
6.16 WFS_EXEE_CIM_INFO_AVAILABLE 84
6.17 WFS_EXEE_CIM_INSERTITEMS 85
6.18 WFS_SRVE_CIM_DEVICEPOSITION 86
6.19 WFS_SRVE_CIM_POWER_SAVE_CHANGE 87

7. ATM Cash-In Transaction Flow - Application Guidelines 88
7.1 OK Transaction (Explicit Shutter Control) 89
7.2 Cancellation by Customer (Explicit Shutter Control) 90
7.3 Stacker Becomes Full (Explicit Shutter Control) 91
7.4 Bill Recognition Error (Explicit Shutter Control) 92
7.5 OK Transaction (Implicit Shutter Control) 93
7.6 Cancellation by Customer (Implicit Shutter Control) 94
7.7 Implicit Control of the Shutter - WFS_EXEE_CIM_SUBCASHIN event 95
7.8 OK Transaction P6 96
7.9 Multiple Refused Notes (Implicit Shutter Control) 97
7.10 Multiple Rollback Notes (Implicit Shutter Control) 99

8. Rules for Cash Unit Exchange 100

9. C - Header file 101

Foreword

This CWA is revision 3.10 of the XFS interface specification.

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

This CWA was formally approved by the XFS Workshop meeting on 2007-11-29. The specification is continuously reviewed and commented in the CEN/ISSS Workshop on XFS. It is therefore expected that an update of the specification will be published in due time as a CWA, superseding this revision 3.10.

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

Parts 19 - 28: Reserved for future use.

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

Part 29: XFS MIB Architecture and SNMP Extensions - 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

Parts 48 - 60 are reserved for future use.

Part 61: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 62: Printer Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 63: Identification Card Device Class Interface - Migration from Version 3.02 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 65: PIN Keypad Device Class Interface - Migration from Version 3.03 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 67: Depository Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.01 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 71: Camera Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 72: Alarm Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 74: Cash-In Module Device Class Interface - Migration from Version 3.02 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

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

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

This CEN Workshop Agreement is publicly available as a reference document from the National Members of CEN : AENOR, AFNOR, ASRO, BDS, BSI, CSNI, CYS, DIN, DS, ELOT, EVS, IBN, IPQ, IST, LVS, LST, MSA, MSZT, NEN, NSAI, ON, PKN, SEE, SIS, SIST, SFS, SN, SNV, SUTN and UNI.

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

Revision History:

3.0	October 18, 2000	First edition
3.02	May 09, 2003	Update release encompassing the Article 6 Paragraph 36 European legislation to deal with handling of forgery and suspected forgery notes. For a detailed description see CWA 14050-28:2003 CIM migration from version 3.0 to version 3.02.
3.10	November 29, 2007	For a description of changes see CWA 15748-74:2007 CIM Migration from Version 3.02 (see CWA 14050) to Version 3.10.

1. Introduction

1.1 Background to Release 3.10

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

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

Release 3.10 of the XFS specification is based on a C API and is delivered with the continued promise for the protection of technical investment for existing applications. This release of the XFS specification has been prompted by a series of factors.

There has been a technical imperative to extend the scope of the existing specification to include new devices, such as the Barcode Reader, Card Dispenser and Item Processing Module.

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

1.2 XFS Service-Specific Programming

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

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

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

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

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

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

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

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

2. Cash-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 (see Section 4.5).

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_RESET
WFS_CMD_CDM_TEST_CASH_UNITS

3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer's Reference Revision 3.10
2. ISO 4217 at http://www.iso.org
3. XFS Cash Dispenser Device Class Interface, Programmer's Reference, Revision 3.10
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

4. Info Commands

4.1 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;
     } 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 or pulling out the device).
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 has detected a fraud attempt.

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 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 door cannot be determined.

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

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

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 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.

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, and optionally one type C.

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

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.

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

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

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.

4.2 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;
     } 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. Normally reflects hardware limitations of the device.

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. This 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. This field will be set to a combination of the following flags:

Value	Meaning
WFS_CIM_RA_RETRACT	Items may be retracted to the retract cash unit.
WFS_CIM_RA_REJECT	Items may be retracted to the 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_NOTSUPP	The CIM does not have the ability to retract.

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

Value	Meaning
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_NOTSUPP	The CIM does not have the ability to retract from the transport.

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

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_NOTSUPP	The CIM does not have the ability to retract from the stacker.

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 for paragraph 6 handling [Ref. 4] is reported in *lpszExtra* as follows:

P6=1	paragraph 6 handling and only level 2 notes will not be returned to the customer in a cash-in transaction.
P6=2	paragraph 6 handling 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.

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

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 a combination of the following flags:

Value	Meaning
WFS_CIM_ITEM_SERIALNUMBER	Serial Number of the item.
WFS_CIM_ITEM_SIGNATURE	P6 Signature of the item.

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

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

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

4.3 WFS_INF_CIM_CASH_UNIT_INFO

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

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

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

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

Counts
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.

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

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

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

Exchanges
If a physical cash unit is inserted (including removal followed by a reinsertion) when the device is not in the exchange state the *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 recycling and retract 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.

recycling cash units (WFS_CIM_TYPERECYCLING) and retract cash units (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 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.
WFS_CIM_CITYPUNFIT	The cash-in unit takes all unfit banknotes.
WFS_CIM_CITYPINDIVIDUAL	The cash-in unit or recycle cash unit takes all types of fit banknotes specified in an individual list.
WFS_CIM_CITYPLEVEL2	All Paragraph 6 level 2 note types are stored in this cash-in unit.
WFS_CIM_CITYPLEVEL3	Paragraph 6 level 3 note types are stored in this cash-in unit.

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.

This value is zero for cash units that cannot accept media items, i.e. cash units that can only dispense.

cUnitID
The Cash Unit Identifier.

cCurrencyID
A three character array storing the ISO format Currency ID [see Ref. 2]. This value will be an array of three ASCII 0x20h characters for cash units which contain items of more than one currency type or items to which currency is not applicable. If the *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 4.5]. If the *cCurrencyID* field for this cash unit is empty 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. For a retract cash unit this value represents the total number of notes of all types in the cash unit, or if the device cannot count notes 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 notes of all types in the cash unit.

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 decremented when these items are either presented to the customer or rejected.

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 may trigger threshold events.

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 (*fwType* = WFS_CIM_TYPECDMSPECIFIC).
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.

lpNoteNumberList
Pointer to a WFSCIMNOTENUMBERLIST structure. The content of this structure is persistent. If the cash unit is a retract cash unit this pointer will be NULL except for the following cases:

If ECB Article 6 is supported and 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.

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.

```
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 items. The value is incremented each time 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 to identify shared cash units.

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.

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 (*fwType* = WFS_CIM_TYPECDMSPECIFIC).
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_STATMANIP	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.

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.

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. This value is persistent. See the CDM interface specification for details.

ulRetractedCount
The number of items that have been retracted into this physical cash unit. This value is persistent.

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

ulRetractedCount
The number of items that have been 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 the reject bin. This value is persistent.

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	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.

Note: WFS_CIM_CITYPALL always overrides WFS_CIM_CITYPINDIVIDUAL when these values are combined.

4.4 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 WFS_INF_CIM_CURRENCY_EXP).

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

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

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

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

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

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

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

Comments None.

4.5 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 [see Ref. 2].

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

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

Comments For each currency ISO 4217 defines the currency identifier (a three character code) and a currency unit (e.g. 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.

4.6 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 [see Ref. 2].

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

usRelease
The release of the banknote type. The higher this number 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.

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

Comments None.

4.7 WFS_INF_CIM_CASH_IN_STATUS

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

Input Param None.

Output Param LPWFSCIMCASHINSTATUS lpCashInStatus;

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

wStatus
Status of the cash-in transaction. Possible values are:

Value	Meaning
WFS_CIM_CIOK	The cash-in transaction is complete.
WFS_CIM_CIROLLBACK	The cash-in transaction was rolled back.
WFS_CIM_CIACTIVE	There is a cash-in transaction active.
WFS_CIM_CIRETRACT	The cash-in transaction ended with the items being retracted.
WFS_CIM_CIUNKNOWN	The state of the cash-in transaction is unknown.
WFS_CIM_CIRESET	The cash-in transaction ended when the WFS_CMD_CIM_RESET command was executed.

usNumOfRefused
Specifies the number of items refused during the cash-in transaction period.

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

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.

4.8 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 level2 / level 3 signatures created. P6 information is available from the point where the WFS_EXEE_CIM_INPUT_P6 event is generated until a command that could move notes within the device is executed or a new cash-in transaction is started.

This command can be used both within and out with a cash-in transaction.

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 x notes. If the pointer is NULL, no level x 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 x 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.

4.9 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 event is generated until a command that could move notes within the device is executed or a new cash-in transaction is started. This command can be used both within and out with a cash-in transaction.

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.

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.

4.10 WFS_INF_CIM_GET_ITEM_INFO

Description This command is used to retrieve the information detected for the items processed during the last command that could move notes. The availability of this information is reported through the WFS_EXEE_CIM_INFO_AVAILABLE event. The data is non-cumulative and is only available until the next command that could move notes is executed (including commands on the CDM interface on recycling devices) or a new cash-in transaction is started. This command can be used both within and out with a cash-in transaction.

The command is similar to the WFS_INF_CIM_GET_P6_SIGNATURE command but returns additional information for Level 2/3 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. The WFS_INF_CIM_GET_ITEM_INFO command (this command) and the WFS_EXEE_CIM_INFO_AVAILABLE apply to every transaction (and WFS_CMD_CIM_CASH_IN in particular). The WFS_EXEE_CIM_INFO_AVAILABLE event signals that item information is available and will be generated during normal transaction processing.

The details about the information available for each note type is reported through the WFS_EXEE_CIM_INFO_AVAILABLE event, 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.

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_2	Information for level 2 notes.
WFS_CIM_LEVEL_3	Information for level 3 notes.
WFS_CIM_LEVEL_4	Information for level 4 notes. This value is also used to retrieve item information on systems that do not support Paragraph 6 classification.

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	P6 Signature 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;
     } WFSCIMITEMINFO, *LPWFSCIMITEMINFO;
```

usNoteID
Identification of note type.

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_CMD_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.

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 to get all item information. 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.

4.11 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;
     } 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. This 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. 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_NOTSUPP	The CIM does not have the ability to retract from this 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.

5. Execute Commands

5.1 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_ROLLBACK, WFS_CMD_CIM_CASH_IN_END, WFS_CMD_CIM_RETRACT or WFS_CMD_CIM_RESET command is sent.

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.

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

fwOutputPosition
The output position where the items will be presented to the customer in the case of a 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.

5.2 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_INPUTITEMS 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 event will be sent to report the rejection.

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

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

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

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.

It is also possible that a device may return refused notes in multiple subsequent bunches. In this case, the WFS_CMD_CIM_CASH_IN command will not complete until the final bunch has been presented and after the last WFS_SRVE_CIM_ITEMSPRESENTED has been generated.

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* parameter. If the whole input was refused then this parameter 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 parameter 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 LPWFSCIMNOTENUMBERLIST structure see the WFS_INF_CIM_CASH_UNIT_INFO command.

The *lpNoteNumberList* contains all notes accepted, if ECB Article 6 is supported then this includes any level 2 or level 3 notes found 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. The cash-in stacker is full.
WFS_ERR_CIM_NOITEMS	There were no items to cash-in.
WFS_ERR_CIM_EXCHANGEACTIVE	The CIM service 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.

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.

Comments None.

5.3 WFS_CMD_CIM_CASH_IN_END

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

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

Input Param None.

Output Param LPWFSCIMCASHINFO lpCashInfo;

lpCashInfo
List of cash units that have taken banknotes or coins and the type of banknotes or coins 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 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_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.

Comments None.

5.4 WFS_CMD_CIM_CASH_IN_ROLLBACK

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

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

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

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 of paragraph 6 handling.

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.

Comments In the special case where all the items inserted by the customer are classified as ECB6 level 2 and/or 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.

5.5 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, 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.

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. Possible values are:

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.

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

Output Param 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 ECB Article 6 is supported and configured). 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	The 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 the retract bin.
WFS_EXEE_CIM_CASHUNITERROR	An error occurred while attempting to retract to the retract bin.
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.

Comments None.

5.6 WFS_CMD_CIM_OPEN_SHUTTER

Description This command opens the shutter.

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 a 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 service 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_ITEMSINSERTED	Items have been inserted by the user.

Comments None.

5.7 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 service is in an exchange state.
WFS_ERR_CIM_SHUTTERNOTCLOSED	Shutter failed to close.
WFS_ERR_CIM_TOOMANYITEMS	There were too many items inserted for the shutter to close.

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

Comments None.

5.8 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.

5.9 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.

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.

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.

5.10 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.

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. Recycling cash units can be set via either interface. However, if the device has recycle units of multiple currencies and/or denominations (or multiple note identifiers associated with the same denomination), then the CIM interface should be used for exchange operations 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_XXX_SET_CASH_UNIT_INFO) 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.

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.

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

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

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

usLogicalNumber
Logical number of recycle unit 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 LPWFSCIMCASHINFO 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 recycler cash unit can not 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.

Comments None.

5.11 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. 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.

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. 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_INVALIDTELLERID	Invalid teller ID. This error will never be generated by a Self-Service CIM.
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_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.

5.12 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.

5.13 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 to the cash unit or output position specified in the *lpResetIn* parameter. This may not always be possible because of hardware problems.

If items are found inside the device 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 LPWFSCIMCAPS structure returned from the WFS_INF_CIM_CAPABILITIES query will determine whether the shutter is controlled implicitly by this command or whether the application must explicitly control the shutter using the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands. 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.

Input Param LPWFSCIMITEMPOSITION lpResetIn;

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

usNumber
The *usNumber* of the cash unit to which items which were inside the CIM when the reset was issued should be moved. If the items should be moved to an output position this value is zero.

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

fwOutputPosition
The output position to which items are to be moved. If the *usNumber* is non-zero then this field will be zero. 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.

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.

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_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.

Comments None.

5.14 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.

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
Type of cash-in unit or recycle unit. Specified as a combination of the following flags:

Value	Meaning
WFS_CIM_CITYPALL	The cash-in unit accepts all fit banknote types.
WFS_CIM_CITYPUNFIT	The cash-in unit accepts all unfit banknotes.
WFS_CIM_CITYPINDIVIDUAL	The cash-in unit or recycle unit accepts all types of fit banknotes specified in the following list.
WFS_CIM_CITYPLEVEL2	All Paragraph 6 level 2 note types are stored in this cash-in unit.
WFS_CIM_CITYPLEVEL3	All Paragraph 6 level 3 note types are stored in this cash-in unit.

See 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 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 service 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.

5.15 WFS_CMD_CIM_CONFIGURE_NOTETYPES

Description This command is used to configure the note types the banknote reader will recognize during cash-in. All note types the banknote reader has to recognize 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.

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

On devices with implicit shutter control, the WFS_EXEE_CIM_INPUTITEMS 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.

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.

Comments None.

5.17 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 and the color. When an application tries to use a color that is not supported then the Service Provider will return the generic error WFS_ERR_UNSUPP_DATA.

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

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

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.

5.18 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 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.

5.19 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 Level2/3 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.

5.20 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.

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. Events

6.1 WFS_SRVE_CIM_SAFEDOOROPEN

Description	This service event specifies that the safe door has been opened.
Event Param	None.
Comments	None.

6.2 WFS_SRVE_CIM_SAFEDOORCLOSED

Description	This service event specifies that the safe door has been closed.
Event Param	None.
Comments	None.

6.3 WFS_USRE_CIM_CASHUNITTHRESHOLD

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

Event Param LPWFSCIMCASHIN lpCashUnit;

lpCashUnit
Pointer to 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.

6.4 WFS_SRVE_CIM_CASHUNITINFOCHANGED

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

WFS_CMD_CIM_SET_CASH_UNIT_INFO
WFS_CMD_CIM_END_EXCHANGE

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

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.

6.5 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.

6.6 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	Specified cash unit is 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 ID 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.

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.

6.7 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.

6.8 WFS_SRVE_CIM_COUNTS_CHANGED

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

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 *usNumbers* of the cash units whose counts have changed.

Comments None.

6.9 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 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.
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 presented is invalid, e.g. it is too large or was presented incorrectly.
WFS_CIM_COUNTERFEIT	One or more counterfeit items have been detected and refused. This is only applicable to devices which do not support ECB Article 6 and are capable of differentiating between invalid and counterfeit items.

Comments None.

6.10 WFS_SRVE_CIM_ITEMSPRESENTED

Description This service event specifies that items have been presented to the output position. In the case of implicit shutter control the items need to be taken. In the case of explicit shutter control the shutter should be 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.

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.

6.11 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.

6.12 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
Specifies 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.

6.13 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 list of banknote numbers which have been identified and accepted during execution of the sub cash-in. This parameter will contain the banknote numbers of the accepted items. For a description of the LPWFSCIMNOTENUMBERLIST structure see the WFS_INF_CIM_CASH_UNIT_INFO command.
Comments	None.

6.14 WFS_SRVE_CIM_MEDIADETECTED

Description	This service event is generated if media is detected during a reset (WFS_CMD_CIM_RESET). The parameter on the event specifies the position of the media on completion of the reset. If the device has been unable to successfully move the items found then this parameter will be NULL.
Event Param	LPWFSCIMITEMPOSITION lpPosition; For a description of this parameter see WFS_CMD_CIM_RESET (section 5.13).
Comments	None.

6.15 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 WFS_INF_CIM_GET_P6_INFO.
Comments	None.

6.16 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_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 at *usLevel* which have information available.

Comments None.

6.17 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.

6.18 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.

6.19 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 None.

7. 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.

	bItemsInsertedSensor==TRUE	*bItemsInsertedSensor*==FALSE
bShutterControl == TRUE	WFS_CMD_CIM_CASH_IN_START WFS_CMD_CIM_CASH_IN InsertedEvent generated WFS_CMD_CIM_CASH_IN_END	WFS_CMD_CIM_CASH_IN_START WFS_CMD_CIM_CASH_IN WFS_CMD_CIM_CASH_IN_END
bShutterControl == FALSE	WFS_CMD_CIM_CASH_IN_START WFS_CMD_CIM_OPEN_SHUTTER InsertedEvent generated WFS_CMD_CIM_CLOSE_SHUTTER WFS_CMD_CIM_CASH_IN WFS_CMD_CIM_CASH_IN_END	WFS_CMD_CIM_CASH_IN_START WFS_CMD_CIM_OPEN_SHUTTER User Input WFS_CMD_CIM_CLOSE_SHUTTER WFS_CMD_CIM_CASH_IN 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.

7.1 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

	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
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
7.			WFS_CMD_CIM_CASH_IN completion of WFS_CMD_CIM_CASH_IN
8.		Display the number of bills and/or amount recognized so far	
9.		Ask the customer for further actions: If they want to insert more money: Repeat from 2. If they want to finish the transaction: Continue with 10. If they want to get back all items inserted so far see table "Cancellation by Customer (Explicit Shutter Control)"	
10.		Transport the money into the cash units RECYCLE_UNIT/CASHINBOX	WFS_CMD_CIM_CASH_IN_END
11.		Credit the money to the customers account	
12.		End of Transaction	

7.2 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

	Customer	**Application**	**XFS Commands and Events**
1.-9.	See OK Transaction (Explicit Shutter Control)		
10.	Selection : Return all the items		
		Transport the items recognized to the output position	WFS_CMD_CIM_CASH_IN_ROLLBACK
11.		Open Shutter	WFS_CMD_CIM_OPEN_SHUTTER
		Request removal of the money.	
	Customer takes the money from the output position		
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
14.		End of Transaction	

7.3 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

	Customer	**Application**	**XFS Commands and Events**
1.-6.	See OK Transaction (Explicit Shutter Control)		
7.			WFS_EXEE_CIM_INPUTREFUSE (StackerFull) and WFS_CMD_CIM_CASH_IN completes with WFS_SUCCESS WFS_SRVE_CIM_ITEMSPRESENTED
8.		Open Shutter	WFS_CMD_CIM_OPEN_SHUTTER
9.		Ask the customer to remove the excess money.	
10.	Customer removes excess money		
11.	If *bItemsTakenSensor* == FALSE Confirm Completion or use application timeout		If *bItemsTakenSensor* == TRUE WFS_SRVE_CIM_ITEMSTAKEN
12.		Close Shutter	WFS_CMD_CIM_CLOSE_SHUTTER
13.		Display the amount recognized so far and tell the customer that the stacker is full	
14.		Ask the customer for further actions: If they want to deposit the amount: Continue with 15. If they want to get back all items inserted so far see table "Cancellation by Customer (Explicit Shutter Control)"	
15.		Transport the money into the cash units RECYCLE_UNIT/CASHINBOX	WFS_CMD_CIM_CASH_IN_END
16.		Ask the customer if they want to deposit more money. If they want to deposit more: Repeat from 1. If they want to finish the transaction: Continue with 17.	
17.		Credit the money to the customers account	
18.		End of Transaction	

7.4 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

	Customer	**Application**	**XFS Commands and Events**
1.-6.	See OK Transaction (Explicit Shutter Control)		
7.			WFS_EXEE_CIM_INPUTREFUSE (InvalidBill) and Completion of WFS_CMD_CIM_CASH_IN with WFS_SUCCESS WFS_SRVE_CIM_ITEMSPRESENTED
8.		Open Shutter	WFS_CMD_CIM_OPEN_SHUTTER
9.		Tell the customer that the bills were not recognized and that he should take the bills.	
10.	Customer removes unrecognized money		
11.	If *bItemsTakenSensor* == FALSE, confirm completion or use application timeout		If *bItemsTakenSensor* == TRUE WFS_SRVE_CIM_ITEMSTAKEN
12.		Close Shutter	WFS_CMD_CIM_CLOSE_SHUTTER
13.		Display the amount recognized so far	
14.		Ask the customer for further actions: If they want to deposit the amount: Continue with 15. If they want to get back all items inserted so far see table "Cancellation by Customer (Explicit Shutter Control)"	
15.		Transport the money into the cash units RECYCLE_UNIT/CASHINBOX	WFS_CMD_CIM_CASH_IN_END
16.		Credit the money to the customers account	
17.		End of Transaction	

7.5 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

	Customer	**Application**	**XFS Commands and Events**
1.	Customer selects cash-in operation.		WFS_CMD_CIM_CASH_IN_START
2.			WFS_CMD_CIM_CASH_IN (Service Provider opens the input shutter). WFS_EXEE_CIM_INPUTITEMS 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
6.			The Service Provider closes the input shutter and begins bill recognition. The WFS_CMD_CIM_CASH_IN command completes.
7.		Display the number of bills and/or amount recognized so far.	
8.		Ask the customer for further actions: If they want to insert more money: Repeat from 2. If they want to finish the transaction: Continue with 9. If they want to get back all items inserted so far see table "Cancellation by Customer (Implicit Shutter Control)"	
9.		Transport the money into the cash units RECYCLE_UNIT/CASHINBOX	WFS_CMD_CIM_CASH_IN_END
10.		Credit the money to the customers account	
11.		End of Transaction	

7.6 Cancellation by Customer (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

	Customer	**Application**	**XFS Commands and Events**
1.-9.	See OK Transaction		
10.	Selection : Return all the items		
11.		Transport the items recognized to the output position	WFS_CMD_CIM_CASH_IN_ROLLBACK.
12.		Request removal of the money.	
13.	Customer takes the money from the output position		
14.	If *bItemsTakenSensor* == FALSE, confirm completion or use application timeout		If *bItemsTakenSensor* == TRUE WFS_SRVE_CIM_ITEMSTAKEN. The Service Provider closes the Shutter.
15.		End of Transaction	

7.7 Implicit Control of the Shutter - WFS_EXEE_CIM_SUBCASHIN event

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 coins 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

	Customer	**Application**	**XFS Commands and Events**
1.-6.	See OK Transaction		
7.			The device processes the bills or coins in batches. Each time a batch is completed a WFS_EXEE_CIM_SUBCASHIN event is posted then the cash-in operation continues.
8.			The WFS_CMD_CIM_CASH_IN command completes.
9.		Display the number of bills and/or amount recognized so far.	
10.		Ask the customer for further actions: If he wants to insert more money: Repeat from 2. If he wants to finish the transaction: Continue with 11. If he wants to get back all items inserted so far see table "Cancellation by Customer (Implicit Shutter Control)"	
11.			WFS_CMD_CIM_CASH_IN_END
12.		End of Transaction	

7.8 OK Transaction P6

This section describes a possible cash-in transaction with P6 where everything works fine and level2 /level 3 notes are inserted.

	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
2.		Ask the customer to insert money	
3.			WFS_CMD_CIM_CLOSE_SHUTTER WFS_CMD_CIM_CASH_IN (WFS_CIM_POSBILLINPUT)
4.	Insert money		WFS_SRVE_CIM_ITEMSINSERTED, WFS_EXEE_CIM_INPUTP6 and completion of WFS_CMD_CIM_CASH_IN
5		Get number of P6 notes	WFS_INF_CIM_GET_P6_INFO
6.		Display the amount recognized so far and inform customer that P6 notes are inserted	
7.		Store signatures of P6 notes with customer data.	Call WFS_INF_CIM_GET_P6_SIGNATURE once for every signature.
8.		Ask the customer for further actions: If customer wants to insert more money: Repeat from 2. If customer wants to finish the transaction: Continue with 9. If 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	WFS_CMD_CIM_CASH_IN_END
10.		At this point the application should decide how to credit the appropriate money to the customers account, and inform the customer about the amounts of level 2 and 3 notes.	
11.		End of Transaction	

7.9 Multiple Refused Notes (Implicit Shutter 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 shutter. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER 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 must be implicit. Therefore, there is no corresponding flow for explicit shutter control.

	Customer	Application	XFS Command
1.-5.	See OK Transaction (Implicit Shutter Control)		
6.			The Service Provider implicitly closes the input shutter and begins bill recognition. As a result of the note processing n batches of notes must be returned to the customer.
7.			WFS_EXEE_CIM_INPUTREFUSE
8.			Return Batch 1 of notes to customer. The Service Provider implicitly opens the shutter. WFS_SRVE_CIM_ITEMSPRESENTED
9.		Tell the customer that the bills were not accepted, and to take the bills.	
10.	Customer removes unrecognized money.		WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter.
11.			Repeat steps 11 through 13 until batches 2 to n-1 are returned to the customer The Service Provider implicitly opens the shutter. WFS_SRVE_CIM_ITEMSPRESENTED
12.		Tell the customer to take the bills	
13.	Customer removes unrecognized money.		WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter.
14.			Return Batch n (last) of notes to customer The Service Provider implicitly opens the shutter. WFS_SRVE_CIM_ITEMSPRESENTED
15.			Completion of WFS_CMD_CIM_CASH_IN with WFS_SUCCESS
16.		Tell the customer to take the bills.	
17.	Customer removes unrecognized money.		
18.			WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter.
19.		Display the amount recognized so far	
20.		Ask the customer for further actions: If they want to deposit the amount: Continue with 21. If they want to get back all items inserted so far see table "Cancellation by Customer (Implicit Shutter Control)"	

21.		Transport the money into the cash units RECYCLE_UNIT/CASHINBOX	WFS_CMD_CIM_CASH_IN_END
22.		Credit the money to the customers account	
23.		End of Transaction	

7.10 Multiple Rollback Notes (Implicit Shutter Control)

The following table describes the flow of a Rollback operation where items are rolled back during the transaction and the Service Provider implicitly controls the shutter. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER 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 must be implicit. Therefore, there is no corresponding flow for explicit shutter control.

	Customer	**Application**	**XFS Command**
1.-10.	See OK Transaction (Implicit Shutter Control)		
		Initiate the rollback operation.	WFS_CMD_CIM_CASH_IN_ROLLBACK
11.			The Service Provider begins the Rollback. As a result of this n batches of notes must be returned to the customer.
12.			Return Batch of notes to customer. The Service Provider implicitly opens the shutter. WFS_SRVE_CIM_ITEMSPRESENTED
13.		Tell the customer to take the bills.	
14.	Customer removes money.		WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter.
15.			Repeat steps 11 through 14 until batches 2 to n-1 are returned to the customer
16.			Return Batch n (last) of notes to customer The Service Provider implicitly opens the shutter. WFS_SRVE_CIM_ITEMSPRESENTED
17.			Completion of WFS_CMD_CIM_CASH_IN_ROLLBACK with WFS_SUCCESS
18.		Tell the customer to take the bills.	
19.	Customer removes money.		
20.			WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter.
21.		End of Transaction	

8. Rules for Cash Unit Exchange

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

If a cash unit is removed from the 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.

9. C - Header file

```
/******************************************************************************
*                                                                             *
* xfscim.h      XFS - Cash Acceptor (CIM) definitions                         *
*                                                                             *
*               Version 3.10 (29/11/2007)                                     *
*                                                                             *
******************************************************************************/

#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       (0x0A03) /* Version 3.10 */
#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)

/* 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)

/* 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)

/* 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

/* 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)

/* 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 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 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)

/* 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)

/* 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 */
```

```
#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 */

#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)

/* values of WFSCIMP6INFO.usLevel */
/* values of WFSCIMP6SIGNATURE.usLevel */

#define     WFS_CIM_LEVEL_2                     (2)
#define     WFS_CIM_LEVEL_3                     (3)
#define     WFS_CIM_LEVEL_4                     (4)

/* 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)

/*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.wItemInfoType */
#define     WFS_CIM_ITEM_SERIALNUMBER          (0x00000001)
#define     WFS_CIM_ITEM_SIGNATURE             (0x00000002)

/* 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)

/* 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 */

#define     WFS_CIM_ADDBUNCHNONE               (1)
#define     WFS_CIM_ADDBUNCHONEMORE            (2)
#define     WFS_CIM_ADDBUNCHUNKNOWN            (3)

/* values of WFSCIMPOSITIONINFO.usBunchesRemaining */

#define     WFS_CIM_NUMBERUNKNOWN              (255)

/* WOSA/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))

/*=================================================================*/
/* CIM Info Command Structures */
/*=================================================================*/

typedef struct _wfs_cim_inpos
{
    WORD                    fwPosition;
    WORD                    fwShutter;
    WORD                    fwPositionStatus;
    WORD                    fwTransport;
    WORD                    fwTransportStatus;
} 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;
} 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;
} 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;
} 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_item_info
{
    USHORT                    usNoteID;
    LPWSTR                    lpszSerialNumber;
    LPWFSCIMP6SIGNATURE       lpP6Signature;
 } 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;
} WFSCIMPOSCAPS, *LPWFSCIMPOSCAPS;

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

/*=================================================================*/
/* 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;

/*=================================================================*/
/* 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;

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

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

#endif  /* __INC_XFSCIM__H */
```