CEN

WORKSHOP

AGREEMENT

CWA 14050-15

October 2003

ICS 35.200; 35.240.40

Supersedes CWA 14050-15:2000

English version

Extensions for Financial Services (XFS) interface specification - Release 3.0 - 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 14050-15:2003 D/E/F

Table of Contents

Foreword

This CWA is revision 3.02 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 document supersedes CWA 14050-15:2000.

This CWA was formally approved by the XFS Workshop meeting on 2003-05-21. 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 3.02.

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

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

Part 2: Service Classes Definition; Programmer's Reference

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

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

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

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

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

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

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

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

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

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

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

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

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

Part 16: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 2.00 (see CWA 13449) to Version 3.00 (this CWA) - Programmer's Reference

Part 17: Printer Device Class Interface - Migration from Version 2.00 (see CWA 13449) to Version 3.00 (this CWA) - Programmer's Reference

Part 18: Identification Card Device Class Interface - Migration from Version 2.00 (see CWA 13449) to Version 3.00 (see CWA 14050-4:2000; superseded) - Programmer's Reference

Part 19: Cash Dispenser Device Class Interface - Migration from Version 2.00 (see CWA 13449) to Version 3.00 (this CWA) - Programmer's Reference

Part 20: PIN Keypad Device Class Interface - Migration from Version 2.00 (see CWA 13449) to Version 3.00 (see CWA 14050-6:2000; superseded) - Programmer's Reference

Part 21: Depository Device Class Interface - Migration from Version 2.00 (see CWA 13449) to Version 3.00 (this CWA) - Programmer's Reference

Part 22: Text Terminal Unit Device Class Interface - Migration from Version 2.00 (see CWA 13449) to Version 3.00 (this CWA) - Programmer's Reference

Part 23: Sensors and Indicators Unit Device Class Interface - Migration from Version 2.00 (see CWA 13449) to Version 3.01 (this CWA) - Programmer's Reference

Part 24: Camera Device Class Interface - Migration from Version 2.00 (see CWA 13449) to Version 3.00 (this CWA) - Programmer's Reference

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

Part 26: Identification Card Device Class Interface - Migration from Version 3.00 (see CWA 14050-4:2000; superseded) to Version 3.02 (this CWA) - Programmer's Reference

Part 27: PIN Keypad Device Class Interface - Migration from Version 3.00 (see CWA 14050-6:2000; superseded) to Version 3.02 (this CWA) - Programmer's Reference

Part 28: Cash In Module Device Class Interface - Migration from Version 3.00 (see CWA 14050-15:2000; superseded) to Version 3.02 (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.cenorm.be/isss/Workshop/XFS.

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

Revision History:

3.00	18 October 2000	First edition
3.02	21 May 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 3.00 to version 3.02

1. Introduction

1.1 Background to Release 3.02

The CEN XFS Workshop is a continuation of the Banking Solution Vendors Council workshop and maintains a technical commitment to the Win 32 API. However, the XFS Workshop has extended the franchise of multi vendor software by encouraging the participation of both banks and vendors to take part in the deliberations of the creation of an industry standard. This move towards opening the participation beyond the BSVC's original membership has been very successful with a current membership level of more than 20 companies.

The fundamental aims of the XFS Workshop are to promote a clear and unambiguous specification for both service providers and application developers. This has been achieved to date by sub groups working electronically and quarterly meetings.

The move from an XFS 3.00 specification to a 3.02 specification has been prompted by customer demand for support of ECB Article 6 legislation to deal with handling of forgery and suspected forgery notes. To do cash recycling in Europe there are requirements defined in article 6 how to deal with money that is a forgery or might be a forgery.
The bank notes are classified in levels. The following levels are defined at the moment:

- level1: no bank note
- level2: forgery
- level3: possibly a forgery
- level4: real money

A signature is a unique identifier for a bank note. It is used together with the transaction data like an account number to identify the customer who has deposited this banknote

The clear direction of the XFS Workshop, therefore, is the delivery of a new Release 3.02 specification based on a C API. It will be delivered with the promise of the protection of technical investment for existing applications and the design to safeguard future developments. All XFS 3.00 CIM clarifications apply to this document.

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 the Extensions for Financial Services 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 behaviour 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 a 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. If the device has recycle units of multiple currencies and/or denominations, then the CIM interface should be used for exchange operations involving these cash units.

The Cash-Out cash unit counts will be available through the CDM interface and the Cash-In cash unit counts will be available through the CIM interface. Counts for recycle cash units are available through both interfaces. The event WFS_SRVE_CIM_COUNTS_CHANGED will be posted if an operation on the CDM interface effects 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 3.00, October 18, 2000
2. ISO 4217 at http://www.iso.ch
3. XFS Cash Dispenser Device Class Interface, Programmer's Reference, Revision 3.00, October 18, 2000
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)

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;
   } WFSCIMSTATUS, * LPWFSCIMSTATUS;
```

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

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.

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

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

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

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

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

4.2 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;
   } WFSCIMCAPS, * LPWFSCIMCAPS;
```

wClass
Supplies the logical service class. Value is:
WFS_SERVICE_CLASS_CIM

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

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 cash in operation. Normally reflects hardware limitations of the device.

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

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

bShutterControl
If set to TRUE the shutter is controlled implicitly by the service provider. If set to FALSE the

shutter must be controlled explicitly by the application using the WFS_CMD_CIM_OPEN_SHUTTER and the WFS_CMD_CIM_CLOSE_SHUTTER commands. This field is always set to TRUE if the device has no shutter. This field applies to all shutters and all output positions.

bSafedoor
Specifies whether the WFS_CMD_CIM_OPEN_SAFE_DOOR command is supported.

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

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

bItemsTakenSensor
Specifies whether or not the CIM can detect when items at the exit position are taken by the user. If set to TRUE the service provider generates an accompanying WFS_SRVE_CIM_ITEMS_TAKEN event. If set to FALSE this event is not generated. This field relates to all output positions.

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

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

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_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 recycle cassettes.
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_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 is 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_NOTSUPP	The CIM does not have the ability to retract from the stacker.

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

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

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 WFSCDMCASHUNIT structure representing the logical cash in unit or recycle unit.

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

Counts
The value of the *ulCount* field of the WFSCIMNOTENUMBER structure is a software count and therefore may not represent the actual number of items in the cash unit.

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

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

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

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

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

Input Param None.

Output Param

```
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;
  } WFSCIMCASHIN, *LPWFSCIMCASHIN;
```

usNumber
Index number of the cash unit structure. Each structure has a unique logical number starting with a value of one (1) for the first structure, and incrementing by one for each subsequent structure.

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

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

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

Value	Meaning
WFS_CIM_CITYPALL	The cash in unit takes all banknote types.
WFS_CIM_CITYPUNFIT	The cash in unit takes all unfit banknotes.
WFS_CIM_CITYPINDIVIDUAL	The cash in unit or recycler takes all types of bank notes specified in an individual list
WFS_CIM_CITYPLEVEL2	Paragraph 6 level 2 notes are stored in this cash in unit
WFS_CIM_CITYPLEVEL3	Paragraph 6 level 3 notes are stored in this cash in unit.

cUnitID
The Cash Unit Identifier.

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

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

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

ulCount
Total number of notes of all types in the cash unit. If the cash unit is a recycle cash unit then this value may not be the same as the value of *ulCashInCount*, the value may be decremented as a result of a dispense operation on the CDM interface. For a retract cash unit this value specifies the number of retracts. This value will be increased by one for every cash in transaction storing level 2 notes. This value is persistent.

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

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

Value	Meaning
WFS_CIM_STATCUOK	The cash unit is in a good state.
WFS_CIM_STATCUFULL	The cash in cash unit or recycle unit is full.
WFS_CIM_STATCUHIGH	The cash in cash unit is almost full (threshold).
WFS_CIM_STATCUEMPTY	The recycle unit is empty.
WFS_CIM_STATCUINOP	The cash in cash unit or recycle unit is inoperative.
WFS_CIM_STATCUMISSING	The cash in 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.
WFS_CIM_STATCUMANIP	The cash unit has been changed 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. If the cash unit is a retract cash unit this pointer will be NULL except when the retract cash unit accepts level 2 notes. In this case the number of level 2 notes is returned.

```
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.

ulCount
Actual count of items. This value is persistent. 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 whenever items leave the cash unit.

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

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

```
typedef struct _wfs_cim_physicalcu
    {
    LPSTR     lpPhysicalPositionName;
    CHAR      cUnitID[5];
    ULONG     ulCashInCount;
    ULONG     ulCount;
    ULONG     ulMaximum;
    USHORT    usPStatus;
    BOOL      bHardwareSensors;
    LPSTR     lpszExtra;
    } 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
Count of items that have entered the cash in unit. This counter is incremented whenever a bill enters the physical cash unit for any reason. This value is persistent.

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

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

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

Value	Meaning
WFS_CIM_STATCUOK	The cash unit is in a good state.
WFS_CIM_STATCUFULL	The cash unit is full.
WFS_CIM_STATCUHIGH	The cash unit is almost full (nearing the threshold defined by ulMaximum).
WFS_CIM_STATCULOW	The cash unit is almost empty (nearing the threshold defined by ulMinimum).
WFS_CIM_STATCUEMPTY	The cash unit is empty.
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.
WFS_CIM_STATCUNOREF	There is no reference value available for the notes in this cash unit. The cash unit has not been configured.
WFS_CIM_STATMANIP	The cash unit has been changed 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
A string of vendor-specific information about the physical cash unit consisting of "*key=value*" sub-strings. Each sub-string is null-terminated, with the final sub-string terminating with two null characters.

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

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

Comments None.

4.4 WFS_INF_CIM_TELLER_INFO

Description This command allows the application to obtain counts for each currency assigned to the teller. It also enables the application to obtain the position assigned to each Teller. If the input parameter is NULL, this command will return information for all Tellers and all currencies. The teller information is persistent.

Input Param

```
LPWFSCIMTELLERINFO  lpTellerInfo;

typedef struct _wfs_cim_teller_info
   {
   USHORT       usTellerID;
   CHAR         cCurrencyID[3];
   } WFSCIMTELLERINFO, *LPWFSCIMTELLERINFO;
```

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

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

Output Param `LPWFSCIMTELLERDETAILS*  lppTellerDetails;`

Pointer to a null-terminated array of pointers to teller info 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 teller total 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 currency exponent 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., German mark, Italian lira). 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 the currency exponent. Thus an amount parameter relates to the actual cash amount as follows:

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

Example #1 — Germany
Currency identifier is 'DEM'
Currency unit is 1 German mark (= 100 pfennig)
A service provider is developed for an ATM that can dispense coins down to one pfennig. The

currency exponent (*sExponent*) is set to -2 (minus two), so the minimum dispense unit is one pfennig (1 * 10^-2 mark); all amounts at the XFS interface are in Pfennig's. Thus a money amount parameter of 10050 is 100 marks and 50 pfennig.

Example #2 — Italy

Currency identifier is 'LIT'

Currency unit is 1 Italian lira

A service provider is required to dispense a minimum amount of 100 lire. The currency exponent (*sExponent*) is set to +2 (plus two), so the minimum dispense unit is 100 lire; all amounts at the XFS interface are in multiples of 100 lire. Thus a amount parameter of 150 is 15000 lire.

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

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

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

Comments None.

4.7 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 lpStatus;`

```
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.

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

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

Comments None.

4.8 WFS_INF_CIM_GET_P6_INFO

Description This command is used to get information about the number of level 2 / level 3 notes on the intermediate stacker and the number of created level2 / level 3 signatures.

Input Param None.

Output Param: `LPWFSCIMP6INFO *lppP6Info`
Pointer to a null terminated array of pointers to p6Info 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 recognised as level x notes. If the pointer is NULL, no level x notes were recognised. 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.

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 (0 to usNumOfLevelxSignatures-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.

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 or WFS_CMD_CIM_CASH_IN_END command is sent.

Input Param

```
LPWFSCIMCASHINSTART lpCashInStart;

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

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

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

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

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.
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.

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.

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

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

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

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

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

Input Param None.

Output Param

```
LPWFSCIMNOTENUMBERLIST   lpNoteNumberList;
```

lpNoteNumberList
List of banknote numbers which have been identified and accepted during execution of this command. If the whole input was refused then this parameter will be NULL and the WFS_EXEE_CIM_INPUTREFUSE event will be generated. If only part of the input was refused then this parameter will contain the banknote numbers of the accepted items and the WFS_EXEE_CIM_INPUTREFUSE event will be generated. For a description of the LPWFSCIMNOTENUMBERLIST structure see the WFS_INF_CIM_CASH_UNIT_INFO command.

The lpNoteNumberList contains all notes accepted including 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 for cash in. 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.
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.

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	A note 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.

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. For a description of the WFSCIMCASHINFO structure see the definition of the WFS_INF_CIM_CASH_UNIT_INFO command.

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

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.

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.

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.

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.

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.
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.

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	Either the items are available to the user or have been removed by the user, depending on the capability of the CIM.

Comments None.

5.5 WFS_CMD_CIM_RETRACT

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

Input Param

```
LPWFSCIMRETRACT     lpRetract;

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.
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_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 the 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 None.

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

Value	Meaning
WFS_ERR_CIM_CASHUNITERROR	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.

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	A note detection error occurred.

Comments None.

5.6 WFS_CMD_CIM_OPEN_SHUTTER

Description This command opens the shutter.

Input Param LPWORD lpfwPosition;

lpfwPosition
Specifies which shutter is to be opened. If the application does not need to specify 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.

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	Either the items are available to the user or have been removed by the user, depending on the capability of the CIM.
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
Specifies which 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.

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

As may the following fields of the WFSCIMPHCU structure:
ulCashInCount
ulCount

Any other changes must be performed via an exchange operation.

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

Input Param
```
LPWFSCIMCASHININFO  lpCUInfo;
```
The `LPWFSCIMCASHININFO` 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_INVALIDTELLERID	Invalid Teller ID.
WFS_ERR_CIM_INVALIDCASHUNIT	Invalid cash unit ID.
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.

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.

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

The command returns current cash unit information in the form described in the documentation of the WFS_INF_CIM_CASH_UNIT_INFO command. This command will also initiate any physical processes which may be necessary to make the cash units accessible. Before using this command an application should first have obtained exclusive control of the CIM.

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

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

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

Any other commands will result in the error WFS_ERR_CIM_EXCHANGEACTIVE being generated

If an error is returned by this command, the WFS_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 must be performed separately on each part of the compound device. These operations cannot be performed simultaneously. An exchange state must therefore be initiated on each interface in the following sequence:

CDM
(Lock)
WFS_CMD_CDM_START_EXCHANGE
…exchange action…
WFS_CMD_CDM_END_EXCHANGE
(Unlock)

CIM
(Lock)
WFS_CMD_CIM_START_EXCHANGE
…exchange action…
WFS_CMD_CIM_END_EXCHANGE
(Unlock)

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

Input Param

```
LPWFSCIMSTARTEX     lpStartEx;

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

f*wExchangeType*
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.
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.

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	An error occurred while performing the exchange operation.
WFS_EXEE_CIM_NOTEERROR	A notes detection error has 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.

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

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

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

Input Param `LPWFSCIMCASHININFO lpCUInfo;`
The `LPWFSCIMCASHININFO` 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.
WFS_ERR_CIM_CASHUNITERROR	This error is returned if there is a problem with the values set for 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	The values of the cash unit structures are incorrect. The cash unit structure that is incorrect is returned as a parameter on this event.

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 nor can it be performed while the CIM is in the exchange state. This command does not end a cash in transaction, the CIM remains in the cash in state.

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

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

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

Input Param `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 0.

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 0. 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 0. 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 this value to NULL. In this case the service provider will determine where to move any items found.

Output Param None.

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

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_EXCHANGEACTIVE	The CIM is in the 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_USRE_CIM_CASHUNITTHRESHOLD	A threshold condition has been reached in one of the cash units.
WFS_EXEE_CIM_CASUNITERROR	A cash unit caused an error.
WFS_SRVE_CIM_MEDIADETECTED	Media was detected during the reset.

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 cash units which are affected by this command must be empty.

The values set by this command are persistent.

Input Param `LPWFSCIMCASHINTYPE * lppCashInType;`

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

```
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 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 bank notes specified in the following list.
WFS_CIM_CITYPLEVEL2	Paragraph 6 level 2 notes are stored in this cash in unit
WFS_CIM_CITYPLEVEL3	Paragraph 6 level 3 notes are stored in this cash in unit

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

Output Param None.

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

Value	Meaning
WFS_ERR_CIM_INVALIDCASHUNIT	Invalid cash unit ID. 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.

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 recognise during cash in. All note types the banknote reader has to recognise must be given in the input structure. If an unknown note type is given the error code WFS_ERR_UNSUPPORTED_DATA will be returned.

The values set by this command are persistent.

Input Param `LPUSHORT lpusNoteIDs;`

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

Output Param None.

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

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.

5.16 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 application may have to execute this command repeatedly to make sure that all possible signatures are captured. If no recognition software is loaded into the recognition module usNoteId will be zero. If the note is not transported to the recognition module (e.g. bad transport out of input position) a NULL pointer is returned.

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 byte.

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 recognised.
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.

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.

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

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 in a denominate or dispense command 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.

Event Param None.

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 a CDM operation.

Event Param

```
LPWFSCIMCOUNTSCHANGED      lpCountsChanged;

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

usCount
The size of lpusCUNumList.

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

Comments None.

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

Value	Meaning
WFS_CIM_CASHINUNITFULL	Cash unit is full.
WFS_CIM_INVALIDBILL	One or more of the items are invalid.
WFS_CIM_NOBILLSTODEPOSIT	There are no bills in the input area.
WFS_CIM_DEPOSITFAILURE	A deposit has failed for a reason other than one of the reasons above, 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.

Comments None.

6.10 WFS_SRVE_CIM_ITEMSPRESENTED

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

Event Param None.

Comments None.

6.11 WFS_SRVE_CIM_ITEMSINSERTED

Description This service event specifies that items have been inserted into the cash in position by the user.

Event Param None.

Comments None.

6.12 WFS_EXEE_CIM_NOTEERROR

Description This execute event specifies the reason for a notes detection error during an operation which involves moving notes.

Event Param `LPUSHORT lpusReason;`

lpusReason
Specifies the reason for the notes 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	A bill counting error has occurred.
WFS_CIM_NOTESTOOCLOSE	Notes have been detected as being too close.

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
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 in operation. (WFS_CMD_CIM_CASH_IN).

Event Param `LPWFSCIMP6INFO *lppP6Info`
Pointer to a null terminated array of pointers to p6Info structures. One structure for every level.

For the description of the structure see WFS_INF_CIM_GET_P6_INFO

Comments None.

7 ATM Cash In Transaction Flow – Application Guidelines

The following table describes the flow of a cash in transaction on a Self Service CIM:

7.1 OK Transaction

This section describes a normal cash in transaction where everything works fine.

	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 and completion of WFS_CMD_CIM_CASH_IN
5.		Display the amount recognized so far	
6.		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 7. If customer wants to get back all items inserted so far see table “cancellation by customer“	
7.		Transport the money into the cash units (RECYCLE_UNIT/CASHINBOX)	WFS_CMD_CIM_CASH_IN_END
8.		Credit the money to the customers account	
9.		End of Transaction	

7.2 Cancellation by Customer

This section describes how an application should react when the customer wants all the items to be returned after recognition.

	Customer	Application	XFS Command
1.-6.	See OK Transaction		
7.	Selection : Return all the items		
8.		Transport the items recognized to the output tray and ask for removal of the money. And inform customer that the P6 notes are stored in the ATM.	WFS_CMD_CIM_CASH_IN_ROLLBACK with output data in case of P6 notes detected WFS_CMD_CIM_OPEN_SHUTTER
9.	Take the money from the output tray		WFS_SRVE_CIM_ITEMSTAKEN
10.		End of Transaction	

7.3 Stacker becomes full

This section describes how an application should react when the stacker becomes full during the transaction.

	Customer	Application	XFS Command
1.-3.	See OK Transaction		
4.	Insert money		WFS_SRVE_CIM_ITEMSINSERTED and completion of WFS_CMD_CIM_CASH_IN with t error code WFS_ERR_CIM_TOOMANYITEMS.
5.		Display the amount recognized so far and tell the customer that the stacker is full	
6.		Ask the customer for further actions: If customer wants to deposit the amount: Continue with 7. If customer wants to get back all items inserted so far see table “cancellation by customer“	
7.		Transport the money into the cash units (RECYCLE_UNIT/CASHINBOX)	WFS_CMD_CIM_CASH_IN_END
8.		Ask the customer if customer wants to deposit more money. If customer wants to deposit more: Repeat from 1. If customer wants to finish the transaction: Continue with 9.	
9.		Credit the money to the customers account	
10.		End of Transaction	

7.4 Bill recognition error

This section describes what an application should do when some of the items could not be recognized (e.g. torn or dirty items) and what sort of interactions with the customer is necessary to complete the transaction.
Please notice that it is only possible to transport the recognized money into the cash in units when the output and the input slot is empty.
So long as the command WFS_CMD_CIM_CASH_IN_END was not issued, the money can be returned to the customer by issuing a WFS_CMD_CIM_CASH_IN_ROLLBACK command. Later returning the money is not longer possible, because it is transported from the stacker to the cash units from where it cannot be taken.

	Customer	**Application**	**XFS Command**
1.-3.	See OK Transaction		
4.	Insert money		WFS_SRVE_CIM_ITEMSINSERTED
5.			WFS_EXEE_CIM_INPUTREFUSE Some of the items could not be recognized (They are moved to the output tray) and completion of WFS_CMD_CIM_CASH_IN
6.			WFS_CMD_CIM_OPEN_SHUTTER
7.		Tell the customer that the bills were not recognized and that customer should take the bills.	
8.	Take the money from the output tray		WFS_SRVE_CIM_ITEMSTAKEN
9.		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 10. If customer wants to get back all items inserted so far see table “cancellation by customer“	
10.		Credit the money to the customers account	
11.		End of Transaction	

7.5 Implicit Control Of the Shutter by the Service Provider – OK Transaction

The following table describes the chronological steps taken in the flow of a Cash In transaction where 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:

	Customer	**Application**	**XFS Command**
1.	Customer selects Cash In operation.		
2.			WFS_CMD_CIM_CASH_IN_START command issued.
3.			The service provider opens the input shutter, then t WFS_CMD_CIM_CASH_IN_START command completes.
4.		Ask customer to insert money into the input shutter then confirm.	
5.	Customer inserts money then confirms.		
6.			WFS_CMD_CIM_CASH_IN command issued.
7.			The service provider closes the input shutter then begins bill recognition. If any bills are not recognized a WFS_EXEE_CIM_INPUT_REFUSED event is posted. The unrecognized notes are returned to the output position and the output shutter is opened. The service provider opens the input shutter on completion for another Cash In operation.
8.			The WFS_CMD_CIM_CASH_IN command completes.
9.		Display number of bills and/or amount recognized and whether any bills were refused. Ask customer if another Cash In operation is required.	
10.	If customer selects another Cash In operation then go to step 4. If customer selects end of Cash In Transaction go to step 11.		
11.			WFS_CMD_CIM_CASH_IN_END command issued.
12.			Service Provider closes the input shutter and if necessary the output shutter.
13.			WFS_CMD_CIM_CASH_IN_END command completes.
14.		End of transaction.	

7.6 Implicit Control Of the Shutter by the Service Provider – RollBack

The following table describes the chronological steps taken in the flow of a Cash In transaction which terminates with a RollBack command. 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:

	Customer	**Application**	**XFS Command**
1.-9.	See OK Transaction		
10.	Customer selects Cancel.		
11.			WFS_CMD_CIM_CASH_IN_ROLLBACK command issued. The Service Provider closes the input shutter and if necessary the output shutter. A notes cashed in since the last WFS_CMD_CIM_CASH_IN_START operation a returned to the customer then the Shutter is opened again to display the bills to the customer.
12.			WFS_CMD_CIM_CASH_IN_ROLLBACK command completes.
13.	Customer takes bills.		
14.			WFS_SRVE_CIM_ITEMSTAKEN event is sent. T 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. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are not used:

	Customer	**Application**	**XFS Command**
1.-6.	See OK Transaction		
7.			The service provider closes the input shutter then begins bill or coin recognition. 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. The service provider opens the input shutter on completion for another Cash In operation.
8.			The WFS_CMD_CIM_CASH_IN command completes.
9.		Display number of bills and/or amount recognized and whether any bills were refused. Ask customer if another Cash In operation is required, if so then go to step 4, otherwise proceed to step 10.	
10.			WFS_CMD_CIM_CASH_IN_END command issued.
11.			Service Provider closes the input shutter and if necessary the output shutter.
12.			WFS_CMD_CIM_CASH_IN_END command completes.
13.		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_EXE_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	

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 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 and possibly tampered with. While the cash unit has this status the Service Provider should not attempt to use it as part of a Dispense operation. The WFS_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 3.02 (09/05/03)                                       *
*                                                                             *
******************************************************************************/

#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       0x0203
#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)

/* 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)


/* 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)


/* 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


/* 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)


/* 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)
```

```
/* 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)

/* values of WFSCIMCASHIN.fwType */

#define     WFS_CIM_TYPERECYCLING               (1)
#define     WFS_CIM_TYPECASHIN                  (2)
#define     WFS_CIM_TYPEREPCONTAINER            (3)
#define     WFS_CIM_TYPERETRACTCASSETTE         (4)

/* 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)
#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)


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

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

/* 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 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)

/* 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)

/* 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))

/*=====================================================================*/
/* 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;
} 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;
} WFSCIMCAPS, * LPWFSCIMCAPS;

typedef struct _wfs_cim_physicalcu
{
    LPSTR           lpPhysicalPositionName;
    CHAR            cUnitID[5];
    ULONG           ulCashInCount;
    ULONG           ulCount;
    ULONG           ulMaximum;
    USHORT          usPStatus;
    BOOL            bHardwareSensors;
    LPSTR           lpszExtra;
} 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;
} 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;


/*=================================================================*/
/* 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_P6_signature
{
    USHORT            usNoteId;
    ULONG             ulLength;
    DWORD             dwOrientation;
    LPVOID            lpSignature;
} WFSCIMP6SIGNATURE, *LPWFSCIMP6SIGNATURE;


/*=================================================================*/
/* CIM Message Structures */
/*=================================================================*/

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

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

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

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

#endif  /* __INC_XFSCIM__H */
```