CEN

WORKSHOP

AGREEMENT

CWA 14050-28

October 2003

ICS 35.200; 35.240.40

English version

Extensions for Financial Services (XFS) interface specification - Release 3.02 - Part 28: Cash In Module Device Class Interface - Migration from Version 3.00 to Version 3.02 - 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 14050-28:2003 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 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.

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 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)

2.0 New Chapters

There are no new chapters

3.0 New Info Commands

3.1 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.

3.2 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.

4.0 Changes to existing Info commands

4.1 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.

4.2 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. 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.

5. New Execute Commands

5.1 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 been taken.
WFS_SRVE_CIM_ITEMSPRESENTED	Items have been presented to the user to be taken.

Comments None.

6.0 Changes to existing Execute commands

6.2 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.

6.3 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.

6.4 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.

7. New Events

7.1 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.

7.0 Changes to existing Events

There are no changes to existing events.

8.0 New ATM Cash-In Transaction Flows

8.1 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	

9.0 Changes to existing ATM Cash In Transaction Flows

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

9.1 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	

9.2 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	

9.3 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	

9.4 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	

9.5 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.	

9.6 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. Al 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. Service Provider closes the Shutter.
15.		End of transaction.	

9.7 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.	

9. C - Header file

```
/******************************************************************************
*                                                                            *
* xfscim.h      XFS - Cash Acceptor (CIM) definitions                        *
*                                                                            *
*               Version 3.02 (08/05/03)                                      *
*                                                                            *
******************************************************************************/

#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 */
```