English version

Extensions for Financial Services (XFS) interface specification
Release 3.20 - Part 15: Cash-In Module Device Class Interface
Programmer's Reference

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties, the constitution of which is indicated in the foreword of this Workshop Agreement.

The formal process followed by the Workshop in the development of this Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN-CENELEC Management Centre can be held accountable for the technical content of this CEN Workshop Agreement or possible conflicts with standards or legislation.

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

This CEN Workshop Agreement is publicly available as a reference document from the CEN Members National Standard Bodies.

CEN members are the national standards bodies of Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and United Kingdom.
## Table of Contents

**Foreword** ....................................................................................................................... 5

1. **Introduction** .......................................................................................................... 8  
   1.1 Background to Release 3.20 .................................................................................. 8  
   1.2 XFS Service-Specific Programming ........................................................................ 8

2. **Cash-In Module** .................................................................................................... 9

3. **References** ......................................................................................................... 10

4. **Legislative Note Handling Standards Support** ................................................ 11

5. **Info Commands** .................................................................................................. 12  
   5.1 WFS_INF_CIM_STATUS ......................................................................................... 12  
   5.2 WFS_INF_CIM_CAPABILITIES .............................................................................. 18  
   5.3 WFS_INF_CIM_CASH_UNIT_INFO ........................................................................ 24  
   5.4 WFS_INF_CIM_TELLER_INFO .............................................................................. 33  
   5.5 WFS_INF_CIM_CURRENCY_EXP ............................................................................ 35  
   5.6 WFS_INF_CIM_BANKNOTE_TYPES ........................................................................ 36  
   5.7 WFS_INF_CIM_CASH_IN_STATUS ......................................................................... 37  
   5.8 WFS_INF_CIM_GET_P6_INFO ............................................................................... 38  
   5.9 WFS_INF_CIM_GET_P6_SIGNATURE ....................................................................... 39  
   5.10 WFS_INF_CIM_GET_ITEM_INFO ......................................................................... 41  
   5.11 WFS_INF_CIM_POSITION_CAPABILITIES .............................................................. 43  
   5.12 WFS_INF_CIM_REPLENISH_TARGET ................................................................... 45  
   5.13 WFS_INF_CIM_DEVICELOCK_STATUS ................................................................. 46  
   5.14 WFS_INF_CIM_CASH_UNIT_CAPABILITIES ........................................................... 47

6. **Execute Commands** .............................................................................................. 49  
   6.1 WFS_CMD_CIM_CASH_IN_START ......................................................................... 49  
   6.2 WFS_CMD_CIM_CASH_IN ..................................................................................... 51  
   6.3 WFS_CMD_CIM_CASH_IN_END ............................................................................ 54  
   6.4 WFS_CMD_CIM_CASH_IN_ROLLBACK .................................................................. 56  
   6.5 WFS_CMD_CIM_RETRACT ..................................................................................... 58  
   6.6 WFS_CMD_CIM_OPEN_SHUTTER ........................................................................... 61  
   6.7 WFS_CMD_CIM_CLOSE_SHUTTER .......................................................................... 62  
   6.8 WFS_CMD_CIM_SET_TELLER_INFO ..................................................................... 63  
   6.9 WFS_CMD_CIM_SET_CASH_UNIT_INFO ................................................................. 64  
   6.10 WFS_CMD_CIM_START_EXCHANGE ................................................................... 66  
   6.11 WFS_CMD_CIM_END_EXCHANGE ....................................................................... 69  
   6.12 WFS_CMD_CIM_OPEN_SAFE_DOOR .................................................................... 70  
   6.13 WFS_CMD_CIM_RESET ....................................................................................... 71
<table>
<thead>
<tr>
<th>Section</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>6.14</td>
<td>WFS_CMD_CIM_CONFIGURE_CASH_IN_UNITS</td>
<td>74</td>
</tr>
<tr>
<td>6.15</td>
<td>WFS_CMD_CIM_CONFIGURE_NOTETYPES</td>
<td>76</td>
</tr>
<tr>
<td>6.16</td>
<td>WFS_CMD_CIM_CREATE_P6_SIGNATURE</td>
<td>77</td>
</tr>
<tr>
<td>6.17</td>
<td>WFS_CMD_CIM_SET_GUIDANCE_LIGHT</td>
<td>80</td>
</tr>
<tr>
<td>6.18</td>
<td>WFS_CMD_CIM_CONFIGURE_NOTE_READER</td>
<td>82</td>
</tr>
<tr>
<td>6.19</td>
<td>WFS_CMD_CIM_COMPARE_P6_SIGNATURE</td>
<td>83</td>
</tr>
<tr>
<td>6.20</td>
<td>WFS_CMD_CIM_POWER_SAVE_CONTROL</td>
<td>85</td>
</tr>
<tr>
<td>6.21</td>
<td>WFS_CMD_CIM_REPLENISH</td>
<td>86</td>
</tr>
<tr>
<td>6.22</td>
<td>WFS_CMD_CIM_SET_CASH_IN_LIMIT</td>
<td>89</td>
</tr>
<tr>
<td>6.23</td>
<td>WFS_CMD_CIM_CASH_UNIT_COUNT</td>
<td>91</td>
</tr>
<tr>
<td>6.24</td>
<td>WFS_CMD_CIM_DEVICE_LOCK_CONTROL</td>
<td>93</td>
</tr>
<tr>
<td>6.25</td>
<td>WFS_CMD_CIM_SET_MODE</td>
<td>96</td>
</tr>
<tr>
<td>6.26</td>
<td>WFS_CMD_CIM_PRESENT_MEDIA</td>
<td>97</td>
</tr>
<tr>
<td>7.1</td>
<td>WFS_SRVE_CIM_SAFEDOOROPEN</td>
<td>99</td>
</tr>
<tr>
<td>7.2</td>
<td>WFS_SRVE_CIM_SAFEDOORCLOSED</td>
<td>100</td>
</tr>
<tr>
<td>7.3</td>
<td>WFS_USRE_CIM_CASHUNITTHRESHOLD</td>
<td>101</td>
</tr>
<tr>
<td>7.4</td>
<td>WFS_SRVE_CIM_CASHUNITINFOCHANGED</td>
<td>102</td>
</tr>
<tr>
<td>7.5</td>
<td>WFS_SRVE_CIM_TELLERINFOCHANGED</td>
<td>103</td>
</tr>
<tr>
<td>7.6</td>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>104</td>
</tr>
<tr>
<td>7.7</td>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>105</td>
</tr>
<tr>
<td>7.8</td>
<td>WFS_SRVE_CIM_COUNTS_CHANGED</td>
<td>106</td>
</tr>
<tr>
<td>7.9</td>
<td>WFS_EXEE_CIM_INPUTREFUSE</td>
<td>107</td>
</tr>
<tr>
<td>7.10</td>
<td>WFS_SRVE_CIM_ITEMSPRESENTED</td>
<td>108</td>
</tr>
<tr>
<td>7.11</td>
<td>WFS_SRVE_CIM_ITEMSSNERTED</td>
<td>109</td>
</tr>
<tr>
<td>7.12</td>
<td>WFS_EXEE_CIM_NOTEERROR</td>
<td>110</td>
</tr>
<tr>
<td>7.13</td>
<td>WFS_EXEE_CIM_SUBCASHIN</td>
<td>111</td>
</tr>
<tr>
<td>7.14</td>
<td>WFS_SRVE_CIM_MEDIADETECTED</td>
<td>112</td>
</tr>
<tr>
<td>7.15</td>
<td>WFS_EXEE_CIM_INPUT_P6</td>
<td>113</td>
</tr>
<tr>
<td>7.16</td>
<td>WFS_EXEE_CIM_INFO_AVAILABLE</td>
<td>114</td>
</tr>
<tr>
<td>7.17</td>
<td>WFS_EXEE_CIM_INSERTITEMS</td>
<td>115</td>
</tr>
<tr>
<td>7.18</td>
<td>WFS_SRVE_CIM_DEVICEPOSITION</td>
<td>116</td>
</tr>
<tr>
<td>7.19</td>
<td>WFS_SRVE_CIM_POWER_SAVE_CHANGE</td>
<td>117</td>
</tr>
<tr>
<td>7.20</td>
<td>WFS_EXEE_CIM_INCOMPLETEREPLENISH</td>
<td>118</td>
</tr>
<tr>
<td>8.1</td>
<td>OK Transaction (Explicit Shutter Control)</td>
<td>119</td>
</tr>
<tr>
<td>8.2</td>
<td>Cancellation by Customer (Explicit Shutter Control)</td>
<td>120</td>
</tr>
<tr>
<td>8.3</td>
<td>Stacker Becomes Full (Explicit Shutter Control)</td>
<td>121</td>
</tr>
<tr>
<td>8.4</td>
<td>Bill Recognition Error (Explicit Shutter Control)</td>
<td>122</td>
</tr>
<tr>
<td>8.5</td>
<td>OK Transaction (Implicit Shutter Control)</td>
<td>123</td>
</tr>
<tr>
<td>8.6</td>
<td>OK Transaction (Implicit Shutter Control)</td>
<td>124</td>
</tr>
</tbody>
</table>
8.6 Cancellation by Customer (Implicit Shutter Control) ......................................................... 125
8.7 Implicit Control of the Shutter - WFS_EXEE_CIM_SUBCASHIN event .......................... 126
8.8 OK Transaction - Note Handling Standard Supported .................................................. 127
8.9 Multiple Refused Notes (Implicit Shutter Control) ......................................................... 128
8.10 Multiple Rollback Notes (Implicit Shutter Control) ...................................................... 130
8.11 Bill Recognition Error (WFS_CMD_CIM_PRESENT_MEDIA Command Supported) ....... 131
8.12 Cancellation by Customer (Implicit Shutter Control and WFS_CMD_CIM_PRESENT_MEDIA Command Supported) .......................................................... 132

9. ATM Mixed Media Transaction Flow – Application Guidelines ................................. 133
  9.1 Mixed Media OK Transaction ..................................................................................... 135
  9.2 Mixed Media Cancellation by Customer ..................................................................... 137
  9.3 Mixed Media Cancellation by Customer on Cash Part Only ....................................... 138
  9.4 Mixed Media Multiple Refused Items ........................................................................ 139

10. Rules for Cash Unit Exchange ..................................................................................... 140

11. C - Header file ............................................................................................................ 142
Foreword

This CWA is revision 3.20 of the XFS interface specification.

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties on 2011-06-29, the constitution of which was supported by CEN following the public call for participation made on 1998-06-24. 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 3.20.

A list of the individuals and organizations which supported the technical consensus represented by the CEN Workshop Agreement is available to purchasers from the CEN-CENELEC Management Centre. These organizations were drawn from the banking sector. The CEN/ISSS XFS Workshop gathered suppliers as well as banks and other financial service companies.

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

Part 1: Application Programming Interface (API) - Service Provider Interface (SPI) - Programmer's Reference
Part 2: Service Classes Definition - Programmer's Reference
Part 3: Printer and Scanning Device Class Interface - Programmer's Reference
Part 4: Identification Card Device Class Interface - Programmer's Reference
Part 5: Cash Dispenser Device Class Interface - Programmer's Reference
Part 6: PIN Keypad Device Class Interface - Programmer's Reference
Part 7: Check Reader/Scanner Device Class Interface - Programmer's Reference
Part 8: Depository Device Class Interface - Programmer's Reference
Part 9: Text Terminal Unit Device Class Interface - Programmer's Reference
Part 10: Sensors and Indicators Unit Device Class Interface - Programmer's Reference
Part 11: Vendor Dependent Mode Device Class Interface - Programmer's Reference
Part 12: Camera Device Class Interface - Programmer's Reference
Part 13: Alarm Device Class Interface - Programmer's Reference
Part 14: Card Embossing Unit Class Interface - Programmer's Reference
Part 15: Cash-In Module Device Class Interface - Programmer's Reference
Part 16: Card Dispenser Device Class Interface - Programmer's Reference
Part 17: Barcode Reader Device Class Interface - Programmer's Reference
Part 18: Item Processing Module Device Class Interface - Programmer's Reference
Parts 19 - 28: Reserved for future use.

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

Part 29: XFS MIB Architecture and SNMP Extensions
Part 30: XFS MIB Device Specific Definitions - Printer Device Class
Part 31: XFS MIB Device Specific Definitions - Identification Card Device Class
Part 32: XFS MIB Device Specific Definitions - Cash Dispenser Device Class
Part 33: XFS MIB Device Specific Definitions - PIN Keypad Device Class
Part 34: XFS MIB Device Specific Definitions - Check Reader/Scanner Device Class
Part 35: XFS MIB Device Specific Definitions - Depository Device Class
Part 36: XFS MIB Device Specific Definitions - Text Terminal Unit Device Class
Part 37: XFS MIB Device Specific Definitions - Sensors and Indicators Unit Device Class
Part 38: XFS MIB Device Specific Definitions - Camera Device Class
CWA 16374-15:2011 (E)

Part 39: XFS MIB Device Specific Definitions - Alarm Device Class
Part 40: XFS MIB Device Specific Definitions - Card Embossing Unit Device Class
Part 41: XFS MIB Device Specific Definitions - Cash-In Module Device Class
Part 42: Reserved for future use.
Part 43: XFS MIB Device Specific Definitions - Vendor Dependent Mode Class
Part 44: XFS MIB Application Management
Part 45: XFS MIB Device Specific Definitions - Card Dispenser Device Class
Part 46: XFS MIB Device Specific Definitions - Barcode Reader Device Class
Part 47: XFS MIB Device Specific Definitions - Item Processing Module Device Class
Parts 48 - 60 are reserved for future use.
Part 61: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 62: Printer and Scanning Device Class Interface - Migration from Version 3.10 (CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 63: Identification Card Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 65: PIN Keypad Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 67: Depository Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 71: Camera Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 72: Alarm Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 74: Cash-In Module Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 75: Card Dispenser Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 76: Barcode Reader Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 77: Item Processing Module Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

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

The formal process followed by the Workshop in the development of the CEN Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN-CENELEC Management Centre can be held accountable for the technical content of the CEN Workshop Agreement or possible conflict with standards or legislation. This CEN Workshop Agreement can in no way be held as being an official standard developed by CEN and its members.

The final review/endorsement round for this CWA was started on 2011-06-23 and was successfully closed on 2011-07-23. The final text of this CWA was submitted to CEN for publication on 2011-08-26.

This CEN Workshop Agreement is publicly available as a reference document from the National Members of CEN: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and the United Kingdom.

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

Revision History:

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.00</td>
<td>October 18, 2000</td>
<td>Initial release.</td>
</tr>
<tr>
<td>3.02</td>
<td>May 09, 2003</td>
<td>For a description of changes from version 3.00 to version 3.02 see the CIM 3.02 Migration document.</td>
</tr>
<tr>
<td>3.10</td>
<td>November 29, 2007</td>
<td>For a description of changes from version 3.02 to version 3.10 see the CIM 3.10 Migration document.</td>
</tr>
<tr>
<td>3.20</td>
<td>March 2nd, 2011</td>
<td>For a description of changes from version 3.10 to version 3.20 see the CIM 3.20 Migration document.</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Background to Release 3.20

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

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

Release 3.20 of the XFS specification is based on a C API and is delivered with the continued promise for the protection of technical investment for existing applications. This release of the specification extends the functionality and capabilities of the existing devices covered by the specification, but does not include any new device classes. Notable major enhancements include Mixed Media processing to allow mixed cash and check accepting, as well as the addition of new commands to the CIM, PTR and IDC to allow better support of the Japanese marketplace.

1.2 XFS Service-Specific Programming

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

When a service-specific command is common among two or more classes of Service Providers, the syntax of the command is as similar as possible across all services, since a major objective of XFS is to standardize function codes and structures for the broadest variety of services. For example, using the \texttt{WFS\_Execute} 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 \texttt{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 \texttt{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 \texttt{WFS\_GetInfo} and \texttt{WFS\_AsyncGetInfo} commands to inquire about the capabilities of the service they are about to use, and modify their behavior accordingly, or they may use functions and then deal with \texttt{WFS\_ERR\_UNSUPP\_COMMAND} error returns to make decisions as to how to use the service.
2. Cash-In Module

This specification describes the functionality of an XFS compliant Cash-In Module (CIM) Service Provider. It defines the service-specific commands that can be issued to the Service Provider using the \texttt{WFSGetInfo}, \texttt{WFSAsyncGetInfo}, \texttt{WFSExecute} and \texttt{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 \texttt{WFS-INF-CIM-CURRENCY-EXP} command.

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

\begin{itemize}
  \item \texttt{WFS-CMD-CIM-SET-TELLER-INFO}
  \item \texttt{WFS-INF-CIM-SET-TELLER-INFO}
\end{itemize}

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 \texttt{WFS-CMD-CIM-END-EXCHANGE} must have completed) before an exchange can start on the other interface. The \texttt{WFS-ERR-CIM-EXCHANGEACTIVE} error code will be returned if the correct sequence is not adhered to.

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

The event \texttt{WFS-SRVE-CIM-COUNTS-CHANGED} will be posted if an operation on the CDM interface affects the recycle cash unit counts which are available through the CIM interface.

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

\begin{itemize}
  \item \texttt{WFS-CMD-CDM-DISPENSE}
  \item \texttt{WFS-CMD-CDM-PRESENT}
  \item \texttt{WFS-CMD-CDM-RETRACT}
  \item \texttt{WFS-CMD-CDM-COUNT}
  \item \texttt{WFS-CMD-CDM-REJECT}
  \item \texttt{WFS-CMD-CDM-SET-CASH-UNIT-INFO}
  \item \texttt{WFS-CMD-CDM-END-EXCHANGE}
  \item \texttt{WFS-CMD-CDM-CALIBRATE-CASH-UNIT}
  \item \texttt{WFS-CMD-CDM-RESET}
  \item \texttt{WFS-CMD-CDM-TEST-CASH-UNITS}
\end{itemize}
3. References

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer’s Reference Revision 3.20</td>
</tr>
<tr>
<td>2.</td>
<td>ISO 4217 at <a href="http://www.iso.org">http://www.iso.org</a></td>
</tr>
<tr>
<td>3.</td>
<td>XFS Cash Dispenser Device Class Interface, Programmer’s Reference, Revision 3.20</td>
</tr>
</tbody>
</table>
4. Legislative Note Handling Standards Support

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

1. Level 1 – Note not recognized. The note is returned to the user.
2. Level 2 – Recognized counterfeit note.
3. Level 3 – Suspected counterfeit note.
4. Level 4 – Recognized note that is identified as genuine.

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

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

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

5.1 WFS_INF_CIM_STATUS

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

Input Param
None.

Output Param
LPWFSCIMSTATUS lpStatus;

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

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

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

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

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

Value Meaning
WFS_CIM_ACCOK All cash units present are in a good state.
WFS_CIM_ACCCUSTATE One or more of the cash units is in a high, full, inoperative or manipulated condition. Items can still be accepted into at least one of the cash units.
WFS_CIM_ACCCUSTOP Due to a cash unit failure accepting is impossible. No items can be accepted because all of the cash units are in a full, inoperative or manipulated condition. This state may also occur when a retract cash unit is full or no retract cash unit is present, or when an application lock is set on every cash unit.
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_NOITEMS The physical device has no intermediate stacker.

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

Value Meaning
WFS_CIM_CUSTOMERACCESS Items on the intermediate stacker have been in customer access. If the device is a cash recycler then the items on the intermediate stacker may be there as a result of a previous cash-out operation.
WFS_CIM_NOCUSTOMERACCESS Items on the intermediate stacker have not been in customer access.
WFS_CIM_ACCESSUNKNOWN It is not known if the items on the intermediate stacker have been in customer access.
WFS_CIM_NOITEMS There are no items on the intermediate stacker or the physical device has no intermediate stacker.

fwBanknoteReader
Supplies the state of the banknote reader as one of the following values:
Value | Meaning
---|---
WFS_CIM_BNROK | The banknote reader is in a good state.
WFS_CIM_BNRINOP | The banknote reader is inoperable.
WFS_CIM_BNRUNKNOWN | Due to a hardware error or other condition, the state of the banknote reader cannot be determined.
WFS_CIM_BNRNOTSUPPORTED | The physical device has no banknote reader.

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

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>Left input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>Right input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>Center input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>Top input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>Bottom input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRONT</td>
<td>Front input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>Rear input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Rear output position.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_SHTCLOSED</td>
<td>The shutter is closed.</td>
</tr>
<tr>
<td>WFS_CIM_SHTOPEN</td>
<td>The shutter is opened.</td>
</tr>
<tr>
<td>WFS_CIM_SHTJAMMED</td>
<td>The shutter is jammed.</td>
</tr>
<tr>
<td>WFS_CIM_SHTUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the shutter cannot be determined.</td>
</tr>
<tr>
<td>WFS_CIM_SHTNOTSUPPORTED</td>
<td>The physical device has no shutter or shutter state reporting is not supported.</td>
</tr>
</tbody>
</table>

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

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

Specifies the state of the transport mechanism as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_TPOK</td>
<td>The transport is in a good state.</td>
</tr>
<tr>
<td>WFS_CIM_TPINOP</td>
<td>The transport is inoperative due to a hardware failure or media jam.</td>
</tr>
<tr>
<td>WFS_CIM_TPKUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the transport cannot be determined.</td>
</tr>
<tr>
<td>WFS_CIM_TPNOTSUPPORTED</td>
<td>The physical device has no transport or transport state reporting is not supported.</td>
</tr>
</tbody>
</table>

Returns information regarding items which may be on the transport. If the device is a cash recycler it is possible that items will be on the transport due to a previous dispense operation, in which case the status will be WFS_CIM_TPSTATNOTEMPTY. The possible values of this field are:

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

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_GUIDANCE_NOT_AVAILABLE</td>
<td>The status is not available.</td>
<td>A</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_OFF</td>
<td>The light is turned off.</td>
<td>A</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_SLOW_FLASH</td>
<td>The light is blinking slowly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_MEDIUM_FLASH</td>
<td>The light is blinking medium frequency.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_QUICK_FLASH</td>
<td>The light is blinking quickly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_CONTINUOUS</td>
<td>The light is turned on continuous (steady).</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_RED</td>
<td>The light is red.</td>
<td>C</td>
</tr>
</tbody>
</table>
wDevicePosition
Specifies the device position. The device position value is independent of the fwDevice value, e.g. when the device position is reported as WFS_CIM_DEVICENOTINPOSITION, fwDevice can have any of the values defined above (including WFS_CIM_DEVONLINE or WFS_CIM_DEVOFFLINE). If the device is not in its normal operating position (i.e. WFS_CIM_DEVICEINPOSITION) then media may not be accepted / presented through the normal customer interface. This value is one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_DEVICEINPOSITION</td>
<td>The device is in its normal operating position, or is fixed in place and cannot be moved.</td>
</tr>
<tr>
<td>WFS_CIM_DEVICENOTINPOSITION</td>
<td>The device has been removed from its normal operating position.</td>
</tr>
<tr>
<td>WFS_CIMDEVICEPOSUNKNOWN</td>
<td>Due to a hardware error or other condition, the position of the device cannot be determined.</td>
</tr>
<tr>
<td>WFS_CIMDEVICEPOSNOTSUPP</td>
<td>The physical device does not have the capability of detecting the position.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_MIXEDMEDIANOTACTIVE</td>
<td>Mixed Media transactions are not supported by the device or Mixed Media mode is not activated.</td>
</tr>
<tr>
<td>WFS_CIM_IPMMIXEDMEDIA</td>
<td>Mixed Media mode using the CIM and IPM interfaces is activated.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_AF MNOTSUPP</td>
<td>No anti-fraud module is available.</td>
</tr>
<tr>
<td>WFS_CIM_AFMOK</td>
<td>Anti-fraud module is in a good state and no foreign device is detected.</td>
</tr>
<tr>
<td>WFS_CIM_AFMINOP</td>
<td>Anti-fraud module is inoperable.</td>
</tr>
<tr>
<td>WFS_CIM_AFMDEVE CITEDECTED</td>
<td>Anti-fraud module detected the presence of a foreign device.</td>
</tr>
<tr>
<td>WFS_CIM_AFMUNKNOWN</td>
<td>The state of the anti-fraud module cannot be determined.</td>
</tr>
</tbody>
</table>

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

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

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

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

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

**Input Param**
None.

**Output Param**
LPWFS_CIMCAPS lpCaps;

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

**wClass**
Specifies the logical service class as WFS_SERVICE_CLASS_CIM.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_TELLERBILL</td>
<td>The CIM is a Teller Bill Acceptor.</td>
</tr>
<tr>
<td>WFS_CIM_SELFSERVICEBILL</td>
<td>The CIM is a Self-Service Bill Acceptor.</td>
</tr>
<tr>
<td>WFS_CIM_TELLERCOIN</td>
<td>The CIM is a Teller Coin Acceptor.</td>
</tr>
<tr>
<td>WFS_CIM_SELFSERVICECOIN</td>
<td>The CIM is a Self-Service Coin Acceptor.</td>
</tr>
</tbody>
</table>

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

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

bShutterControl
If set to TRUE the shutter is controlled implicitly by the Service Provider. If set to FALSE the shutter must be controlled explicitly by the application using the WFS_CMD_CIM_OPEN_SHUTTER and the WFS_CMD_CIM_CLOSE_SHUTTER commands. In either case the WFS_CMD_CIM_PRESENT_MEDIA command may be used if the bPresentControl field is reported as FALSE. The bShutterControl field is always set to TRUE if the device has no shutter. This field applies to all shutters and all positions.

bSafeDoor
Specifies whether the WFS_CMD_CIM_OPEN_SAFE_DOOR command is supported.

bCashBox
This field is only applicable to CIM types WFS_CIM_TELLERBILL and WFS_CIM_TELLERCOIN. It specifies whether or not the tellers have been assigned a cash box.

bRefill
This field is not used.

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

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

bItemsInsertedSensor
Specifies whether the CIM has the ability to detect when items have actually been inserted by the user. If set to TRUE the Service Provider generates an accompanying WFS_SRVE_CIM_ITEMSINSERTED event. If set to FALSE this event is not generated. This field relates to all input positions. This flag should not be reported as TRUE unless item insertion can be detected.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>Left input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>Right input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>Center input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>Top input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>Bottom input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRONT</td>
<td>Front input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>Rear input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Rear output position.</td>
</tr>
</tbody>
</table>
fwExchangeType
Specifies the type of cash unit exchange operations supported by the CIM. Values are a combination of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_EXBYHAND</td>
<td>The CIM supports manual replenishment either by emptying the cash unit by hand or by replacing the cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_EXTOCASSETTES</td>
<td>The CIM supports moving items from the replenishment cash unit to the bill cash units.</td>
</tr>
<tr>
<td>WFS_CIM_CLEARRECYCLER</td>
<td>The CIM supports the emptying of recycle cash units.</td>
</tr>
<tr>
<td>WFS_CIM_DEPOSITINTO</td>
<td>The CIM supports moving items from the deposit entrance to the bill cash units.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_RA_RETRACT</td>
<td>Items may be retracted to a retract cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_RA_REJECT</td>
<td>Items may be retracted to a reject cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_RA_TRANSPORT</td>
<td>Items may be retracted to the transport.</td>
</tr>
<tr>
<td>WFS_CIM_RA_STACKER</td>
<td>Items may be retracted to the intermediate stacker.</td>
</tr>
<tr>
<td>WFS_CIM_RA_BILLCASSETTES</td>
<td>Items may be retracted to item cassettes, i.e. cash-in and recycle cash units.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_PRESENT</td>
<td>The items may be moved to the exit position.</td>
</tr>
<tr>
<td>WFS_CIM_RETRACT</td>
<td>The items may be retracted to a retract cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_REJECT</td>
<td>The items may be retracted to a reject cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_BILLCASSETTES</td>
<td>The items may be retracted to item cassettes, i.e. cash-in and recycle cash units.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_PRESENT</td>
<td>The items may be moved to the exit position.</td>
</tr>
<tr>
<td>WFS_CIM_RETRACT</td>
<td>The items may be retracted to a retract cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_REJECT</td>
<td>The items may be retracted to a reject cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_BILLCASSETTES</td>
<td>The items may be retracted to item cassettes, i.e. cash-in and recycle cash units.</td>
</tr>
</tbody>
</table>

lpszExtra
Pointer to a list of vendor-specific, or any other extended, information. The information is returned as a series of “key=value” strings so that it is easily extensible by Service Providers. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.
The parameter that reports if a legislative note handling standard is supported is reported in
\textit{lpszExtra} as follows:

\begin{itemize}
\item \textbf{P6=1} A note handling standard is supported and only level 2 notes will not be returned to the customer in a cash-in transaction.
\item \textbf{P6=2} A note handling standard is supported and level 2 and level 3 notes will not be returned to the customer in a cash-in transaction.
\end{itemize}

\textit{dwGuidLights [...]} Specifies which guidance light positions are available. A number of guidance light positions are defined below. Vendor specific guidance lights are defined starting from the end of the array. The maximum guidance light index is \texttt{WFS\_CIM\_GUIDLIGHTS\_MAX}.

The elements of this array are specified as a combination of the following flags and indicate all of the possible flash rates (type B) and colors (type C) that the guidance light indicator is capable of handling. If the guidance light indicator only supports one color then no value of type C is returned. A value of \texttt{WFS\_CIM\_GUIDANCE\_NOT\_AVAILABLE} indicates that the device has no guidance light indicator or the device controls the light directly with no application control possible.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_NOT_AVAILABLE}</td>
<td>There is no guidance light control available at this position.</td>
<td>A</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_OFF}</td>
<td>The light can be off.</td>
<td>B</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_SLOW_FLASH}</td>
<td>The light can blink slowly.</td>
<td>B</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_MEDIUM_FLASH}</td>
<td>The light can blink medium frequency.</td>
<td>B</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_QUICK_FLASH}</td>
<td>The light can blink quickly.</td>
<td>B</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_CONTINUOUS}</td>
<td>The light can be continuous (steady).</td>
<td>B</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_RED}</td>
<td>The light can be red.</td>
<td>C</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_GREEN}</td>
<td>The light can be green.</td>
<td>C</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_YELLOW}</td>
<td>The light can be yellow.</td>
<td>C</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_BLUE}</td>
<td>The light can be blue.</td>
<td>C</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_CYAN}</td>
<td>The light can be cyan.</td>
<td>C</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_MAGENTA}</td>
<td>The light can be magenta.</td>
<td>C</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_WHITE}</td>
<td>The light can be white.</td>
<td>C</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSINNULL}</td>
<td>The default input position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSINLEFT}</td>
<td>Left input position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSINRIGHT}</td>
<td>Right input position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSINCENTER}</td>
<td>Center input position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSINTOP}</td>
<td>Top input position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSINBOTTOM}</td>
<td>Bottom input position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSINFRONT}</td>
<td>Front input position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSINREAR}</td>
<td>Rear input position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSOUTLEFT}</td>
<td>Left output position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSOUTRIGHT}</td>
<td>Right output position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSOUTCENTER}</td>
<td>Center output position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSOUTTOP}</td>
<td>Top output position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSOUTBOTTOM}</td>
<td>Bottom output position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSOUTFRONT}</td>
<td>Front output position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSOUTREAR}</td>
<td>Rear output position.</td>
</tr>
<tr>
<td>\texttt{WFS_CIM_GUIDANCE_POSOUTNULL}</td>
<td>The default output position.</td>
</tr>
</tbody>
</table>

\textit{dwItemInfoTypes} Specifies the types of information that can be retrieved through the \texttt{WFS\_INF\_CIM\_GET\_ITEM\_INFO} command as a combination of the following flags:
### Value Meaning

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_ITEM_SERIALNUMBER</td>
<td>Serial Number of the item.</td>
</tr>
<tr>
<td>WFS_CIM_ITEM_SIGNATURE</td>
<td>Signature of the item.</td>
</tr>
</tbody>
</table>

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

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_LIMITBYTOTALITEMS</td>
<td>The number of successfully processed cash-in items can be limited by specifying the total number of items.</td>
</tr>
<tr>
<td>WFS_CIM_LIMITBYAMOUNT</td>
<td>The number of successfully processed cash-in items can be limited by specifying the total amount.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_COUNTINDIVIDUAL</td>
<td>The counting of individual cash units via the input structure of the <code>WFS_CMD_CIM_CASH_UNIT_COUNT</code> command is supported.</td>
</tr>
<tr>
<td>WFS_CIM_COUNTALL</td>
<td>The counting of all cash units via the NULL pointer input parameter of the <code>WFS_CMD_CIM_CASH_UNIT_COUNT</code> command is supported.</td>
</tr>
</tbody>
</table>

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

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

*bMixedDepositAndRollback*

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

*bAntiFraudModule*

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

**Error Codes**

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

**Comments**

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

<table>
<thead>
<tr>
<th><em>bShutter</em></th>
<th><em>bShutterControl</em></th>
<th><em>WFSCIMPOSCAPS</em>. <em>bPresentControl</em></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>Service Provider implicitly opens the shutter, presents items and closes the shutter when all items are taken.</td>
</tr>
<tr>
<td>TRUE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>Service Provider implicitly opens the shutter for input. Application required to present items using WFS_CMD_CIM_PRESENT_MEDIA.</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>Application is required to present items using WFS_CMD_CIM_OPEN_SHUTTER and then call WFS_CMD_CIM_CLOSE_SHUTTER when all items are taken.</td>
</tr>
<tr>
<td>TRUE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>Application required to present items using WFS_CMD_CIM_PRESENT_MEDIA, or alternatively, by using, WFS_CMD_CIM_OPEN_SHUTTER and then WFS_CMD_CIM_CLOSE_SHUTTER when all items are taken.</td>
</tr>
<tr>
<td>FALSE</td>
<td>TRUE</td>
<td>TRUE</td>
<td>Service Provider implicitly opens the shutter, presents items and closes the shutter when all items taken.</td>
</tr>
<tr>
<td>FALSE</td>
<td>TRUE</td>
<td>FALSE</td>
<td>Service Provider implicitly opens the shutter for input. Application required to present items using WFS_CMD_CIM_PRESENT_MEDIA.</td>
</tr>
<tr>
<td>FALSE</td>
<td>FALSE</td>
<td>TRUE</td>
<td>Not Supported.</td>
</tr>
<tr>
<td>FALSE</td>
<td>FALSE</td>
<td>FALSE</td>
<td>Application required to present items using WFS_CMD_CIM_PRESENT_MEDIA.</td>
</tr>
</tbody>
</table>
5.3 WFS_INF_CIM_CASH_UNIT_INFO

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

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

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

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

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

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

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

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

For a cash unit of type WFS_CIM_TYPERETRACTCASSETTE, it is also possible that the threshold event can instead be triggered by the ulCashInCount reaching the ulMaximum value. For more detail see the bRetractNoteCountThresholds field description in the WFS_INF_CIM_CASH_UNIT_CAPABILITIES command.

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

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

Exchanges
If a physical cash unit is inserted (including removal followed by a reinsertion) when the device is not in the exchange state the usPStatus of the physical cash unit will be set to WFS_CIM_STATCUMANIP and the values of the physical cash unit prior to its’ removal will be returned in any subsequent WFS_INF_CIM_CASH_UNIT_INFO command. The physical cash unit will not be used in any operation. The application must perform an exchange operation specifying the new values for the physical cash unit in order to recover the situation.
On recycle and retract cash units the counts and status reflect the physical status of the cassette and therefore are consistently reported on both the CDM and CIM interfaces. When a value is changed through an exchange on one interface it is also changed on the other.

Recyclers

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

Input Param

None.

Output Param

LPWFSCIMCASHINFO lpCashInfo;

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

usCount

Number of WFSCIMCASHIN structures returned in lppCashIn.

lppCashIn

Pointer to an array of pointers to WFSCIMCASHIN structures:

typedef struct _wfs_cim_cash_in
{
    USHORT    usNumber;
    DWORD     fwType;
    DWORD     fwItemType;
    CHAR     cUnitID[5];
    CHAR     cCurrencyID[3];
    ULONG     ulValues;
    ULONG     ulCashInCount;
    ULONG     ulCount;
    ULONG     ulMaximum;
    USHORT    usStatus;
    BOOL     bAppLock;
    LPWFSCIMNOTENUMBERLIST lpNoteNumberList;
    USHORT    usNumPhysicalCUs;
    LPWFSCIMPHCU   *lppPhysical;
    LPSTR     lpszExtra;
    LPUSHORT    lpusNoteIDs;
    WORD     usCDMType;
    LPSTR     lpszCashUnitName;
    ULONG     ulInitialCount;
    ULONG     ulDispensedCount;
    ULONG     ulPresentedCount;
    ULONG     ulRetractedCount;
    ULONG     ulRejectCount;
    ULONG     ulMinimum;
} WFSCIMCASHIN, *LPWFSCIMCASHIN;

usNumber

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

fwType

Specifies the type of cash unit as one of the following values:

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

**WFS_CIM_TYPERETRACTCASSETTE**  
Retract cash unit.

**WFS_CIM_TYPEREJECT**  
Reject cash unit.

**WFS_CIM_TYPECDMSPECIFIC**  
A cash unit that is only applicable to the CDM interface. This value is used to report CDM cash units of the following types: WFS_CDM_TYPENA, WFS_CDM_TYPEBILLCASSETTE, WFS_CDM_TYPECOINCYLINDER, WFS_CDM_TYPECOINDISPENSER, WFS_CDM_TYPECOUPON and WFS_CDM_TYPEDOCUMENT. See the usCDMType field for details of the cash unit type.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_CITYPALL</td>
<td>The cash-in unit takes all fit banknote types. If a note handling standard is supported, then these are level 4 notes which are fit for recycling.</td>
</tr>
<tr>
<td>WFS_CIM_CITYPUNFIT</td>
<td>The cash-in unit takes all unfit banknotes. If a note handling standard is supported, then these are level 4 notes which are unfit for recycling.</td>
</tr>
<tr>
<td>WFS_CIM_CITYPINDIVIDUAL</td>
<td>The cash-in unit or recycle cash unit takes all types of fit banknotes specified in an individual list. If a note handling standard is supported, then these are level 4 notes which are fit for recycling.</td>
</tr>
<tr>
<td>WFS_CIM_CITYPLEVEL2</td>
<td>If a note handling standard is supported, then level 2 note types are stored in this cash-in unit.</td>
</tr>
<tr>
<td>WFS_CIM_CITYPLEVEL3</td>
<td>If a note handling standard is supported, then level 3 note types are stored in this cash-in unit.</td>
</tr>
<tr>
<td>WFS_CIM_CITYPIPM</td>
<td>The cash-in unit can accept items on the IPM interface.</td>
</tr>
</tbody>
</table>

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

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

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

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

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

**ulCashInCount**
Count of items that have entered the logical cash unit. This counter is incremented whenever an item enters a physical cash unit that belongs to this logical cash unit for any reason. For a retract cash unit this value represents the total number of items of all types in the cash unit, or if the device cannot count items during a retract operation this value will be zero. If *fwType* is `WFS_CIM_TYPECDMSPECIFIC` then this value is zero. This value is persistent.

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

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>WFS_CIM_STATCUOK</code></td>
<td>The cash unit is in a good state.</td>
</tr>
<tr>
<td><code>WFS_CIM_STATCUFULL</code></td>
<td>The cash unit is full. This value is not used for CDM specific cash units (fwType == <code>WFS_CIM_TYPECDMSPECIFIC</code>).</td>
</tr>
</tbody>
</table>
WFS_CIM_STATCUHIGH  The cash unit is almost full (i.e. reached or exceeded the threshold defined by ulMaximum). This value is not used for CDM specific cash units (fwType == WFS_CIM_TYPECDMSPECIFIC).

WFS_CIM_STATCULOW  The cash unit is almost empty (i.e. reached or below the threshold defined by ulMinimum). This value is only reported for CDM specific cash units (fwType == WFS_CIM_TYPECDMSPECIFIC).

WFS_CIM_STATCUEMPTY  The cash unit is empty. On a dispensing cash unit on a recycler this can be caused by insufficient items in the cash unit preventing further dispense operations.

WFS_CIM_STATCUNOP  The cash unit is inoperative.

WFS_CIM_STATCUMISSING  The cash unit is missing.

WFS_CIM_STATCUNOVAL  The values of the specified cash unit are not available. This can be the case when the cash unit is changed without using the operator functions.

WFS_CIM_STATCUNOREF  There is no reference value available for the notes in this cash unit. The cash unit has not been configured. This value has no meaning on the CIM and is not used.

WFS_CIM_STATCUMANIP  The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state. Items cannot be accepted into this cash unit.

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

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

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

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

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

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

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

If both cases apply then the number and type of level 2 notes and notes retracted is returned in the lpNoteNumberList and ulCount contains the number of retract operations. ulCashInCount contains the actual number of level 2 notes and retracted items.
typedef struct _wfs_cim_note_number_list
{
    USHORT usNumOfNoteNumbers;
    LPWFS_CIMNOTENUMBER *lppNoteNumber;
} WFS_CIMNOTENUMBERLIST, *LPWFS_CIMNOTENUMBERLIST;

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 WFS_CIMNOTENUMBER structures:

typedef struct _wfs_cim_note_number
{
    USHORT usNoteID;
    ULONG ulCount;
} WFS_CIMNOTENUMBER, *LPWFS_CIMNOTENUMBER;

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

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

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

lppPhysical
Pointer to an array of pointers to WFS_CIMPHCU structures:

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

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

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

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

ulCount
As defined by the logical ulCount description but applies to a single physical cash unit. The one exception is that during dispense transactions on the CDM, this value does not include any items that have been dispensed but not yet presented. This value is persistent.
Maximum count of items in the physical cash unit. No threshold event will be generated when this value is reached. This value is persistent. This field is deprecated. The value for \texttt{ulMaximum} is reported using the \texttt{WFS-INF-CIM-CASH-UNIT-CAPABILITIES} command.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{WFS-CIM-STATCUOK}</td>
<td>The cash unit is in a good state.</td>
</tr>
<tr>
<td>\texttt{WFS-CIM-STATCUFULL}</td>
<td>The cash unit is full. This value is not used for CDM specific cash units (\texttt{fwType \neq WFS-CIM_TYPECDMSPECIFIC}).</td>
</tr>
<tr>
<td>\texttt{WFS-CIM-STATCUHIGH}</td>
<td>The cash unit is almost full (reached or exceeded the threshold defined by \texttt{ulMaximum} in physical structure). This value is not used for CDM specific cash units (\texttt{fwType \neq WFS-CIM_TYPECDMSPECIFIC}).</td>
</tr>
<tr>
<td>\texttt{WFS-CIM-STATCULOW}</td>
<td>The cash unit is almost empty. This value is only reported for CDM specific cash units (\texttt{fwType \neq WFS-CIM_TYPECDMSPECIFIC}).</td>
</tr>
<tr>
<td>\texttt{WFS-CIM-STATCUEMPTY}</td>
<td>The cash unit is empty. On a dispensing cash unit on a recycler this can be caused by insufficient items in the cash unit preventing further dispense operations.</td>
</tr>
<tr>
<td>\texttt{WFS-CIM-STATCUINOP}</td>
<td>The cash unit is inoperative.</td>
</tr>
<tr>
<td>\texttt{WFS-CIM-STATCUMISSING}</td>
<td>The cash unit is missing (the cash unit has been removed and is physically not present in the machine).</td>
</tr>
<tr>
<td>\texttt{WFS-CIM-STATCUNOVAL}</td>
<td>The values of the specified cash unit are not available.</td>
</tr>
<tr>
<td>\texttt{WFS-CIM-STATCUNOREF}</td>
<td>There is no reference value available for the notes in this cash unit. The cash unit has not been configured. This value is only reported for CDM specific cash units (\texttt{fwType \neq WFS-CIM_TYPECDMSPECIFIC}).</td>
</tr>
<tr>
<td>\texttt{WFS-CIM-STATCUMANIP}</td>
<td>The cash unit has been inserted (including removal followed by a reinsertion) when the device was not in the exchange state.</td>
</tr>
</tbody>
</table>

Specifies whether or not threshold events can be generated based on hardware sensors in the device. If this value is \texttt{TRUE} for any of the physical cash units related to a logical cash unit then threshold events may be generated based on hardware sensors as opposed to logical counts. This field is deprecated. The value for \texttt{ulMaximum} is reported using the \texttt{WFS-INF-CIM-CASH-UNIT-CAPABILITIES} command.

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

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

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

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

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

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

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

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

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

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

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

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

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

ulRejectCount
The number of items from this logical cash unit which are in a reject bin. This value is persistent.

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

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

Comments
The following table defines the interpretation of the fwItemType flag for single values and a subset of possible combinations (many of which may not actually be possible on physical hardware implementations). The check mark means that the corresponding flag is set, empty means that the corresponding flag is not set.
For a definition of the terms 'fit' and 'unfit' see the description of `fwItemType` itself. The combinations not included in this table can be interpolated from this table.

<table>
<thead>
<tr>
<th>ALL</th>
<th>UNFIT</th>
<th>INDIVIDUAL</th>
<th>LEVEL 3</th>
<th>LEVEL 2</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td>Fit notes for all note ids</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td>Unfit notes for all note ids</td>
</tr>
<tr>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>Fit notes from the Individual note list</td>
</tr>
<tr>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td>Level 3 notes for all note ids</td>
</tr>
<tr>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>Level 2 notes for all note ids</td>
</tr>
<tr>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td>Fit notes for all note ids &amp; unfit notes for all note ids</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td>Fit notes for all note ids &amp; level 3 notes for all note ids</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fit notes for all note ids &amp; level 2 notes for all note ids</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td>Fit notes for all note ids &amp; level 3 notes for all note ids &amp; level 2 notes for all note ids</td>
</tr>
<tr>
<td>√</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td>Fit notes from the Individual note list &amp; unfit notes for all note ids</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td>Fit notes from the Individual note list &amp; level 3 notes for all note ids</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Fit notes from the Individual note list &amp; level 2 notes for all note ids</td>
</tr>
<tr>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td>Fit notes from the Individual note list &amp; level 3 notes for all note ids &amp; level 2 notes for all note ids</td>
</tr>
</tbody>
</table>

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

WFS_CIM_CITYPIPM can be combined with any other combination and indicates non-note items can be stored in this cash unit.
5.4 WFS_INF_CIM_TELLER_INFO

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

Input Param
LPWFSCIMTELLERINFO lpTellerInfo;

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

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

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

This parameter can be an array of three ASCII 0x20 characters. In this case information on all currencies will be returned.

Output Param
LPWFSCIMTELLERDETAILS *lppTellerDetails;

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

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

usTellerID
Identification of teller.

fwInputPosition
The input position assigned to the teller for cash entry. The value is set to one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSNULL</td>
<td>No position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>The left position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>The right position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>The center position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>The top position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>The bottom position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRONT</td>
<td>The front position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>The rear position is assigned to the teller.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSNULL</td>
<td>No position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>The left position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>The right position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>The center position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>The top position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>The bottom position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>The front position is assigned to the teller.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>The rear position is assigned to the teller.</td>
</tr>
</tbody>
</table>
typedef struct _wfs_cim_teller_totals
{
    CHAR     cCurrencyID[3];
    ULONG     ulItemsReceived;
    ULONG     ulItemsDispensed;
    ULONG     ulCoinsReceived;
    ULONG     ulCoinsDispensed;
    ULONG     ulCashBoxReceived;
    ULONG     ulCashBoxDispensed;
} WFSCIMTELLERTOTALS, *LPWFSCIMTELLERTOTALS;

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

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_INVALIDCURRENCY</td>
<td>Specified currency not currently available.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_INVALIDTELLERID</td>
<td>Invalid teller ID.</td>
</tr>
</tbody>
</table>

Comments
None.
5.5 WFS_INF_CIM_CURRENCY_EXP

Description
This command returns each exponent assigned to each currency known to the Service Provider.

Input Param
None.

Output Param
LPWFS_CIM_CURRENCY_EXP *lpplppCurrencyExp;
Pointer to a NULL-terminated array of pointers to WFS_CIM_CURRENCY_EXP structures:

typedef struct _wfs_cim_currency_exp
{
    CHAR     cCurrencyID[3];
    SHORT     sExponent;
} WFS_CIM_CURRENCY_EXP, *LPWFS_CIM_CURRENCY_EXP;

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

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

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

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

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

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

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

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

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

Description

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

Input Param

None.

Output Param

LPWFSIMNOTETYPELIST lpNoteTypeList;

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

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

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

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

usNoteID
Identification of note type.

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

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

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

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

Error Codes

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

Comments

None.
5.7  WFS-INF-CIM-CASH-IN-STATUS

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

**Input Param**
None.

**Output Param**
LPWFSCIMCASHEINSTATUS lpCashInStatus;

```c
typedef struct _wfs_cim_cash_in_status {
    WORD     wStatus;     // Status of the cash-in transaction. Possible values are:
    USHORT   usNumOfRefused;  // Specifies the number of items refused during the cash-in transaction period.
    LPWFSCIMNOTENUMBERLIST lpNoteNumberList;  // List of banknote types that were inserted, identified and accepted during the cash-in transaction period. The WFSCIMNOTENUMBERLIST.ulCount value within this structure is the count of items of identified and accepted notes during the cash-in transaction period. If notes have been rolled back they will be included in this list. If wStatus is WFS_CIM_CIRETRACT only identified and accepted notes are included in this list. For a description of the WFSCIMNOTENUMBERLIST structure see the definition of the command WFS-INF-CIM-CASH-UNIT-INFO.
    LPSTR     lpszExtra;    // Pointer to a list of vendor-specific, or any other extended, information. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.
} WFSCIMCASHEINSTATUS, *LPWFSCIMCASHEINSTATUS;
```

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

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

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

**lpNoteNumberList**
List of banknote types that were inserted, identified and accepted during the cash-in transaction period. The WFSCIMNOTENUMBERLIST.ulCount value within this structure is the count of items of identified and accepted notes during the cash-in transaction period. If notes have been rolled back they will be included in this list. If wStatus is WFS_CIM_CIRETRACT only identified and accepted notes are included in this list. For a description of the WFSCIMNOTENUMBERLIST structure see the definition of the command WFS-INF-CIM-CASH-UNIT-INFO.

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

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

**Comments**
None.
5.8 **WFS_INF_CIM_GET_P6_INFO**

**Description**

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

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

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

- WFS_CMD_CDM_DISPENSE, WFS_CMD_CDM_COUNT, WFS_CMD_CDM_PRESENT,
- WFS_CMD_CDM_RETRACT, WFS_CMD_CDM_REJECT,
- WFS_CMD_CDM_OPEN_SHUTTER, WFS_CMD_CDM_CLOSE_SHUTTER,
- WFS_CMD_CIM_RESET, WFS_CMD_CDM_START_EXCHANGE,
- WFS_CMD_CIM_END_EXCHANGE, WFS_CMD_CDM_CALIBRATE_CASH_UNIT,
- WFS_CMD_CDM_TEST_CASH_UNITS.

**Input Param**

None.

**Output Param**

LPWFSLOBAL_CIMINFO *lppP6Info;

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

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

- **usLevel**
  
  Defines the note level. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_LEVEL_2</td>
<td>Information for level 2 notes.</td>
</tr>
<tr>
<td>WFS_CIM_LEVEL_3</td>
<td>Information for level 3 notes.</td>
</tr>
</tbody>
</table>

- **lpNoteNumberList**

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

- **usNumOfSignatures**

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

**Error Codes**

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

**Comments**

None.
5.9 WFS-INF-CIM-GET-P6-SIGNATURE

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

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

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

WFS_CMD_CDM_DISPENSE, WFS_CMD_CDM_COUNT, WFS_CMD_CDM_PRESENT,
WFS_CMD_CDM_RETRACT, WFS_CMD_CDM_REJECT,
WFS_CMD_CDM_OPEN_SHUTTER, WFS_CMD_CDM_CLOSE_SHUTTER,
WFS_CMD_CIM_RESET, WFS_CMD_CIM_START_EXCHANGE,
WFS_CMD_CIM_END_EXCHANGE, WFS_CMD_CIM_CALIBRATE_CASH_UNIT,
WFS_CMD_CIM_TEST_CASH_UNITS.

This command is used to retrieve the required information on an individual item basis.
Applications should loop retrieving the information for each index and for each level reported
with the WFS-INF-CIM-GET-P6-INFO command.

Input Param

LPWFSCIMGETP6SIGNATURE lpGetP6Signature;

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_LEVEL_2</td>
<td>The application wants a level 2 signature.</td>
</tr>
<tr>
<td>WFS_CIM_LEVEL_3</td>
<td>The application wants a level 3 signature.</td>
</tr>
</tbody>
</table>

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

Output Param

LPWFSCIMP6SIGNATURE lpP6Signature;

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

usNoteId
Identification of note type.

ulLength
Length of the signature in bytes.

dwOrientation
Orientation of the entered banknote. Specified as one of the following flags:
<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_ORFRONNTTOP</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_CIM_ORFRONTBOTTOM</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_CIM_ORBACKTOP</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_CIM_ORBACKBOTTOM</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_CIM_ORUNKNOWN</td>
<td>The orientation for the inserted note can not be determined.</td>
</tr>
<tr>
<td>WFS_CIM_ORNOTSUPPORTED</td>
<td>The hardware is not capable to determine the orientation.</td>
</tr>
</tbody>
</table>

\[ lpSignature \]
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.
5.10 WFS_INF_CIM_GET_ITEM_INFO

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

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

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

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

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

Input Param
LPWFSFCIMGETITEMINFO lp.GetItemInfo;

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

usLevel
Defines the note level. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_LEVEL_2</td>
<td>Information for level 2 notes.</td>
</tr>
<tr>
<td>WFS_CIM_LEVEL_3</td>
<td>Information for level 3 notes.</td>
</tr>
<tr>
<td>WFS_CIM_LEVEL_4</td>
<td>Information for level 4 notes. This value is</td>
</tr>
<tr>
<td></td>
<td>also used to retrieve item information on</td>
</tr>
<tr>
<td></td>
<td>systems that do not support note handling</td>
</tr>
<tr>
<td></td>
<td>standards.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_ITEM_SERIALNUMBER</td>
<td>Serial Number of the item.</td>
</tr>
<tr>
<td>WFS_CIM_ITEM_SIGNATURE</td>
<td>Signature of the item.</td>
</tr>
</tbody>
</table>

Output Param
LPWFSFCIMITEMINFO lpItemInfo;

The data returned by this command relates to a single item (usIndex).
typedef struct _wfs_cim_item_info
{
    USHORT    usNoteID;
    LPWSTR    lpszSerialNumber;
    LPWFSCEM6SIGNATURE lpP6Signature;
} WFSCIMITEMINFO, *LPWFSCIMITEMINFO;

usNoteID
Identification of note type.

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

lpP6Signature
This field contains the signature for the item, see the WFS_CMD_CIM_GET_P6_SIGNATURE command for a description of the contents. If no signature is available or has not been requested then this field is NULL.

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

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

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

**Input Param**
None.

**Output Param**
LPWFSIMPOSCAPABILITIES lpPosCaps;

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

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>Left input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>Right input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>Center input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>Top input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>Bottom input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRONT</td>
<td>Front input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>Rear input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Rear output position.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSIN</td>
<td>It is an input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSREFUSE</td>
<td>It is an output position used to refuse items.</td>
</tr>
<tr>
<td>WFS_CIM_POSROLLBACK</td>
<td>It is an output position used to rollback items.</td>
</tr>
</tbody>
</table>

**bShutterControl**
If set to TRUE the shutter is controlled implicitly by the Service Provider. If set to FALSE the shutter must be controlled explicitly by the application using the WFS_CMD_CIM_OPEN_SHUTTER and the WFS_CMD_CIM_CLOSE_SHUTTER commands. In either case the WFS_CMD_CIM_PRESENT_MEDIA command may be used if the bPresentControl field is reported as FALSE. The bShutterControl field is always set to TRUE if the described position has no shutter.
bItemsTakenSensor
Specifies whether or not the described position can detect when items at the exit position are taken by the user. If set to TRUE the Service Provider generates an accompanying WFS_SRVE_CIM_ITEMSTAKEN event. If set to FALSE this event is not generated. This field relates to output and refused positions.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_RA_RETRACT</td>
<td>Items may be retracted to a retract cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_RA_REJECT</td>
<td>Items may be retracted to a reject cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_RA_TRANSPORT</td>
<td>Items may be retracted to the transport.</td>
</tr>
<tr>
<td>WFS_CIM_RA_STACKER</td>
<td>Items may be retracted to the intermediate stacker.</td>
</tr>
<tr>
<td>WFS_CIM_RA_BILLCASSETTES</td>
<td>Items may be retracted to item cassettes, i.e. cash-in and recycle cash units.</td>
</tr>
</tbody>
</table>

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

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

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

Comments
None.
5.12 WFS_INF_CIM_REPLENISH_TARGET

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

Input Param
LPWFSCIMREPINFO lpReplenishInfo;

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

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

Output Param
LPWFSCIMREPINFOReres lpReplenishInfoResult;

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

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

typedef struct _wfs_cim_info_target {
    USHORT usNumberTarget;
} WFSCIMREPINFOTARGET, *LPWFSCIMREPINFOTARGET;

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

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

Comments
None.
5.13 WFS_INF_CIMDEVICELOCK_STATUS

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

Input Param
None.

Output Param
LPWFS_CIMDEVICELOCKSTATUS lpDevLockStatus;

typedef struct _wfs_cim_device_lock_status
{  
    WORD wDeviceLockStatus;
    LPWFS_CIMCASHUNITLOCK *lppCashUnitLock;
} WFS_CIMDEVICELOCKSTATUS, *LPWFS_CIMDEVICELOCKSTATUS;

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

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

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

typedef struct _wfs_cim_cash_unit_lock
{  
    LPSTR lpPhysicalPositionName;
    WORD wCashUnitLockStatus;
} WFS_CIMCASHUNITLOCK, *LPWFS_CIMCASHUNITLOCK;

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

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

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

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

Comments
None.
5.14 WFS_INF_CIM_CASH_UNIT_CAPABILITIES

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

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

Input Param
None.

Output Param
LPWFSCIMCASHCAPABILITIES lpCashCaps;

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

usCount
Number of WFSCIMCASHUNITCAPABILITIES structures returned in lppCashUnitCaps.

lppCashUnitCaps
Pointer to an array of pointers to WFSCIMCASHUNITCAPABILITIES structures:

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

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

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

lppPhysical
Pointer to an array of pointers to WFSCIMPHCUCAPABILITIES structures:

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

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

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

bHardwareSensors
Specifies whether or not threshold events can be generated based on hardware sensors in the device. If this value is TRUE for any of the physical cash units related to a logical cash unit then threshold events may be generated based on hardware sensors as opposed to logical counts.
lpszExtra
Pointer to a list of vendor-specific information about the physical cash unit. The information is returned as a series of "key=value" strings so that it is easily extensible by Service Providers. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.

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

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

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

Comments
None.
6. Execute Commands

6.1 WFS_CMD_CIM_CASH_IN_START

Description
Before initiating a cash-in operation, an application must issue the WFS_CMD_CIM_CASH_IN_START command to begin a cash-in transaction. During a cash-in transaction any number of WFS_CMD_CIM_CASH_IN commands may be issued. The transaction is ended when either a WFS_CMD_CIM_CASH_IN_ROLLBACK, WFS_CMD_CIM_CASH_IN_END, WFS_CMD_CIM_RETRACT or WFS_CMD_CIM_RESET command is sent.

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

Input Param
LPWFS_CIM_CASH_IN_START lpCashInStart;

typedef struct _wfs_cim_cash_in_start
{
    USHORT usTellerID;
    BOOL bUseRecycleUnits;
    WORD fwOutputPosition;
    WORD fwInputPosition;
} WFS_CIM_CASH_IN_START, *LPWFS_CIM_CASH_IN_START;

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSNULL</td>
<td>The items will be presented to the default configuration.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>The items will be presented to the left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>The items will be presented to the right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>The items will be presented to the center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>The items will be presented to the top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>The items will be presented to the bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>The items will be presented to the front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>The items will be presented to the rear output position.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSNULL</td>
<td>The cash is inserted from the default configuration.</td>
</tr>
</tbody>
</table>
CWA 16374-15:2011 (E)

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:

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

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

Comments  None.
6.2 WFS_CMD_CIM_CASH_IN

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

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

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

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

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

The bShutterControl field of the WFSCIMCAPS structure returned from the WFS_INF_CIM_CAPABILITIES query will determine whether the shutter is controlled implicitly by this command or whether the application must explicitly open and close the shutter using the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands, or the WFS_CMD_CIM_PRESENT_MEDIA command. If bShutterControl is FALSE then this command does not operate the shutter in any way, the application is responsible for all shutter control. If bShutterControl is TRUE this command opens the shutter at the start of the command and closes it once bills are inserted.

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

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

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

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

Input Param
None.

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_CASHUNITERROR</td>
<td>A problem occurred with a cash unit. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_TOOMANYITEMS</td>
<td>There were too many items inserted previously. The cash-in stacker is full at the beginning of this command.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_NOITEMS</td>
<td>There were no items to cash-in.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SHUTTERNOTCLOSED</td>
<td>Shutter failed to close. In the case of explicit shutter control the application should close the shutter first.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_NOCASINACTIVE</td>
<td>There is no cash-in transaction active.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_POSITION_NOT_EMPTY</td>
<td>The output position is not empty so a cash-in is not possible.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SAFEDOOROPEN</td>
<td>The safe door is open. This device requires the safe door to be closed in order to perform a WFS_CMD_CIM_CASH_IN command.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_FOREIGN_ITEMS_DETECTED</td>
<td>Foreign items have been detected inside the input position.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SHUTTERNOTOPEN</td>
<td>Shutter failed to open.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>A problem occurred with a cash unit.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INPUT_P6</td>
<td>Level 2 and / or level 3 notes are detected.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INPUTREFUSE</td>
<td>A part or all of the amount of the cash-in order was refused.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_NOTEERROR</td>
<td>An item detection error occurred.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_SUBCASIN</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSINSERTED</td>
<td>Items have been inserted into the cash-in position by the user.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>The items have been removed by the user. This event is only generated if the bItemsTakenSensor field returned in the capabilities information is TRUE.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSPRESENTED</td>
<td>Items have been presented to the user to be taken.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INFO_AVAILABLE</td>
<td>Information is available for items detected during the cash processing operation.</td>
</tr>
<tr>
<td>Event Name</td>
<td>Description</td>
</tr>
<tr>
<td>----------------------------</td>
<td>-----------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INSERTITEMS</td>
<td>Device is ready to accept items from the user.</td>
</tr>
<tr>
<td>WFS_USRE_CIM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has occurred in one of the cash units.</td>
</tr>
</tbody>
</table>

**Comments**
None.
6.3 WFS_CMD_CIM_CASH_IN_END

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

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

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

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

Input Param
None.

Output Param
LPWFS_CIM_CASHINFO lpCashInfo;

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

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CIM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has occurred in one of the cash units.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_CASHUNITINFOCHANGED</td>
<td>A cash unit was changed.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>A problem occurred with the cash unit.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INPUT_P6</td>
<td>Level 2 and / or level 3 notes are detected during this operation.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INFOAVAILABLE</td>
<td>Information is available for items detected during the cash processing operation.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_NOTEERROR</td>
<td>An item detection error occurred.</td>
</tr>
<tr>
<td>Event Code</td>
<td>Description</td>
</tr>
<tr>
<td>------------------------------------</td>
<td>--------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>The items have been removed by the user. This event is only generated during a Mixed Media transaction where the IPM items are presented and taken and the WFSCIMCAPS.blItemsTakenSensor field is TRUE.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSPRESENTED</td>
<td>Items have been presented to the user to be taken. This event is only generated during a Mixed Media transaction where the IPM items are presented.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_COUNTS_CHANGED</td>
<td>In Mixed Media mode, counters can be changed by the command WFS_CDM_IPM_MEDIA_IN_END.</td>
</tr>
</tbody>
</table>

**Comments**

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

Description

This command is used to roll back a cash-in transaction. It causes all the cash items cashed in since the last WFS_CMD_CIM_CASH_IN_START command to be returned to the customer. This command ends the current cash-in transaction. The cash-in transaction is ended even if this command does not complete successfully.

The bShutterControl field of the WFSCIMCAPS structure returned from the WFS_INF_CIM_CAPABILITIES query will determine whether the shutter is controlled implicitly by this command or whether the application must explicitly control the shutter using the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands, or WFS_CMD_CIM_PRESENT_MEDIA command. If bShutterControl is FALSE then this command does not operate the shutter in any way, the application is responsible for all shutter control. If bShutterControl is TRUE then this command opens the shutter and it is closed when all items are removed.

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

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

Input Param

NULL.

Output Param

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

LPWFSCIMCASHINFO lpCashInfo;

lpCashInfo

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

Error Codes

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_CASHUNITERROR</td>
<td>A problem occurred with a cash unit. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SHUTTERNOTOPEN</td>
<td>Shutter failed to open. In the case of explicit shutter control the application may have failed to open the shutter before issuing the command.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in the exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_NOCASHINACTION</td>
<td>There is no current cash-in transaction.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_POSITION_NOT_EMPTY</td>
<td>The input or output position is not empty.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_NOITEMS</td>
<td>There were no items to rollback.</td>
</tr>
</tbody>
</table>

Events

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

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

Comments

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

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

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

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

Input Param
LPWFCIMRETRACT lpRetract;

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

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_RA_RETRACT</td>
<td>Retract the items to a retract cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_RA_REJECT</td>
<td>Retract the items to a reject cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_RA_TRANSPORT</td>
<td>Retract the items to the transport.</td>
</tr>
<tr>
<td>WFS_CIM_RA_STACKER</td>
<td>Retract the items to the intermediate stacker area.</td>
</tr>
<tr>
<td>WFS_CIM_RA_BILLCASSETTES</td>
<td>Retract the items to item cassettes, i.e. cash-in and recycle cash units.</td>
</tr>
</tbody>
</table>
If `usRetractArea` is set to WFS_CIM_RA_RETRACT this field defines the position inside the retract cash units into which the cash is to be retracted. `usIndex` starts with a value of one (1) for the first retract position and increments by one for each subsequent position. If there are several logical retract cash units (of type WFS_CIM_TYPERETRACTCASSETTE in command WFS_INF_CIM_CASH_UNIT_INFO), `usIndex` would be incremented from the first position of the first retract cash unit to the last position of the last retract cash unit defined in WFSCIMCASHINFO. The maximum value of `usIndex` is the sum of the `ulMaximum` of each retract cash unit. If `usRetractArea` is not set to WFS_CIM_RA_RETRACT the value of this field is ignored.

**Output Param**

LPWFSCIMCASHINFO lpCashInfo;

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

**Error Codes**

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

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

**Events**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CIM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in a retract bin.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>An error occurred while attempting to retract to a retract bin.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_NOTEERROR</td>
<td>An item detection error occurred.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INPUT_P6</td>
<td>Level 2 and / or level 3 notes are detected during this operation.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>The items have been removed by the user. This event is only generated if the <code>bItemsTakenSensor</code> field returned in the capabilities information is TRUE.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INFO_AVAILABLE</td>
<td>Information is available for items detected during the cash processing operation.</td>
</tr>
<tr>
<td><strong>WFS_SRVE_CIM_CASHUNITINFOCHANGED</strong></td>
<td>A cash unit was updated as a result of this command.</td>
</tr>
<tr>
<td><strong>Comments</strong></td>
<td>None.</td>
</tr>
</tbody>
</table>
6.6 WFS_CMD_CIM_OPEN_SHUTTER

**Description**
This command opens the shutter.

**Input Param**
LPWORD lpfwPosition;

*lpfwPosition*
Pointer to the position where the shutter is to be opened. If the application does not need to specify the shutter, this field can be set to NULL or to WFS_CIM_POSNULL. Otherwise this field should be set to one of the following values:

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

**Output Param**
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SHUTTERNOTOPEN</td>
<td>Shutter failed to open.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SHUTTEROPEN</td>
<td>Shutter was already open.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_FOREIGN_ITEMSDET</td>
<td>Foreign items have been detected in the input position.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>The items have been removed by the user. This event is only generated if the bItemsTakenSensor field returned in the capabilities information is TRUE.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSINSERTED</td>
<td>Items have been inserted by the user.</td>
</tr>
</tbody>
</table>

**Comments**
None.
6.7 WFS_CMD_CIM_CLOSE_SHUTTER

**Description**
This command closes the shutter.

**Input Param**
LPWORD lpfwPosition;

*lpfwPosition*
Pointer to the position where the shutter is to be closed. If the application does not need to specify the shutter, this field can be set to NULL or to WFS_CIM_POSNULL. Otherwise this field should be set to one of the following values:

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

**Output Param**
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SHUTTERCLOSED</td>
<td>Shutter was already closed.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SHUTTERNOTCLOSED</td>
<td>Shutter failed to close.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_TOOMANYITEMS</td>
<td>There were too many items inserted for the shutter to close.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_FOREIGN_ITEMS_DETECTED</td>
<td>Foreign items have been detected in the input position. The shutter is open.</td>
</tr>
</tbody>
</table>

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

**Comments**
None.
6.8 WFS_CMD_CIM_SET_TELLER_INFO

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

**Input Param**
LPWFSCIMTELLERUPDATE lpTellerUpdate;

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_CREATE_TELLER</td>
<td>A teller is to be added.</td>
</tr>
<tr>
<td>WFS_CIM_MODIFY_TELLER</td>
<td>Information about an existing teller is to be modified.</td>
</tr>
<tr>
<td>WFS_CIM_DELETE_TELLER</td>
<td>A teller is to be removed.</td>
</tr>
</tbody>
</table>

lpTellerDetails
For a specification of the 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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_INVALIDCURRENCY</td>
<td>The specified currency is not currently available.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_INVALIDTELLERID</td>
<td>The teller ID is invalid.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The target teller is currently in the middle of an exchange operation.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_SRVE_CIM_TELLERINFOCHANGED</td>
<td>Teller information has been created, modified or deleted.</td>
</tr>
</tbody>
</table>

**Comments**
None.
6.9 WFS_CMD_CIM_SET_CASH_UNIT_INFO

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

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

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

The following fields of the WFSCIMCASHIN structure may be updated by this command:

- ulCount
- ulCashInCount
- ulMaximum
- bAppLock
- lpNoteNumberList (contents must be consistent with ulCount)
- ulInitialCount
- ulDispensedCount
- ulPresentedCount
- ulRetractedCount
- ulRejectCount
- ulMinimum

As may the following fields of the WFSCIMPHCU structure:

- ulCashInCount
- ulCount
- ulInitialCount
- ulDispensedCount
- ulPresentedCount
- ulRetractedCount
- ulRejectCount

Any other changes must be performed via an exchange operation.

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

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

Input Param
LPWFSCIMCASHINFO lpCUInfo;

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

Output Param
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_INVALIDCASHUNIT</td>
<td>Invalid cash unit.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_CASHUNITERROR</td>
<td>A problem occurred with a cash unit. A WFS_EXEE_CIM_CASHUNITERROR event will be posted with the details.</td>
</tr>
</tbody>
</table>

Events
In addition to the generic events defined in [Ref. 1], the following events can be generated as a result of this command:
<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CIM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_CASHUNITINFOCHANGED</td>
<td>A cash unit was updated as a result of this command.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>An error occurred while accessing a cash unit.</td>
</tr>
</tbody>
</table>

**Comments**  
None.
6.10 WFS_CMD_CIM_START_EXCHANGE

Description

This command puts the CIM in an exchange state, i.e. a state in which cash units can be emptied, replenished, removed or replaced. Other than the updates which can be made via the WFS_CMD_CIM_SET_CASH_UNIT_INFO command all changes to a cash unit must take place while the cash unit is in an exchange state.

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

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

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

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

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

If the CIM is part of a compound device together with a CDM (i.e. a cash recycler), exchange operations can either be performed separately on each interface to the compound device, or the entire exchange operation can be done through the CIM interface.

Exchange via CDM and CIM interfaces

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

CDM

(Lock)

WFS_CMD_CDM_START_EXCHANGE

…exchange action…

WFS_CMD_CDM_END_EXCHANGE

(Unlock)

CIM

(Lock)

WFS_CMD_CIM_START_EXCHANGE

…exchange action…

WFS_CMD_CIM_END_EXCHANGE

(Unlock)

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

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

**Input Param**

```
LPWFSCIMSTARTEX lpStartEx;
```

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

**fwExchangeType**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_EXBYHAND</td>
<td>The cash units will be replenished manually either by filling or emptying the cash unit by hand or by replacing the cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_EXTOCASSETTES</td>
<td>Items will be moved from the replenishment container to the bill cash units. Items will be moved from the bill cash units to the replenishment container. On a cash recycler, the CDM interface should be used to move items from a replenishment container.</td>
</tr>
<tr>
<td>WFS_CIM_CLEARRECYCLER</td>
<td>Items will be moved from a recycle cash unit to a cash unit or output position.</td>
</tr>
<tr>
<td>WFS_CIM_DEPOSITINTO</td>
<td>Items will be moved from the deposit entrance to the bill cash units.</td>
</tr>
</tbody>
</table>

**usTellerID**

Identification of teller. If the device is a Self-Service CIM this field is ignored.

**usCount**

Number of cash units to be exchanged. This is also the size of the array contained in the `lpusCUNumList` field.

**lpusCUNumList**

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

**lpOutput**

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

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

**usLogicalNumber**

Logical number of recycle cash unit be emptied.

**fwPosition**

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

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

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

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

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>A cash unit caused an error.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_NOTEERROR</td>
<td>An item detection error occurred.</td>
</tr>
<tr>
<td>WFS_USRE_CIM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has occurred in one of the cash units. This event is not generated for recycle cash units.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_CASHUNITINFOCHANGED</td>
<td>A cash unit was changed.</td>
</tr>
</tbody>
</table>

**Comments**
None.
6.11 WFS_CMD_CIM_END_EXCHANGE

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

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

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

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

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

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

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

Input Param
LPWFSCIMCASHINFO lpCUInfo;

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

Output Param
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_CASHUNITERROR</td>
<td>A cash unit problem occurred that meant no cash units could be updated. One or more WFS_EXEE_CIM_CASHUNITERROR events will be sent with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_NOEXCHANGEACTIVE</td>
<td>There is no exchange active.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CIM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_CASHUNITINFOCHANGED</td>
<td>A cash unit was changed.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>A cash unit caused an error.</td>
</tr>
</tbody>
</table>

Comments
None.
6.12 WFS_CMD_CIM_OPEN_SAFE_DOOR

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

**Input Param**
None.

**Output Param**
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in an exchange state.</td>
</tr>
</tbody>
</table>

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

**Comments**
None.
6.13 WFS_CMD_CIM_RESET

**Description**

This command is used by the application to perform a hardware reset which will attempt to return the CIM device to a known good state. This command does not over-ride a lock obtained on another application or service handle.

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

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

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

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

The `bShutterControl` field of the WFSCIMCAPS structure returned from the WFS_INF_CIM_CAPABILITIES query will determine whether the shutter is controlled implicitly by this command or whether the application must explicitly control the shutter using the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands, or the WFS_CMD_CIM_PRESENT_MEDIA command. If `bShutterControl` is FALSE then this command does not operate the shutter in any way, the application is responsible for all shutter control. If `bShutterControl` is TRUE then this command operates the shutter as necessary so that the shutter is closed after the command completes successfully and any items returned to the customer have been removed.

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

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

**Input Param**

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

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

**usNumber**

In the case of a single cash unit destination this value specifies the cash unit to be used for the storage of any items found, i.e. when items are to be moved to a reject or retract cash unit. In all other cases this value must be zero, i.e. when items are to be moved to item cassettes, the transport, the stacker or an output position.

**lpRetractArea**

This field is used if items are to be moved to the stacker, the transport, a retract cassette or to item cassettes. If items are to be moved to a reject cash unit or to an output position then this field must be NULL.
typedef struct _wfs_cim_retract {
    WORD     fwOutputPosition;
    USHORT   usRetractArea;
    USHORT   usIndex;
} WFSCIMRETRACT, *LPWFSCIMRETRACT;

fwOutputPosition
This value will be ignored.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_RA_RETRACT</td>
<td>Items will be moved to a retract cash unit. In the case where several cash units of type WFS_CIM_TYPERETRACTCASSETTE exist the usNumber field will define which retract unit the items will be moved to.</td>
</tr>
<tr>
<td>WFS_CIM_RA_TRANSPORT</td>
<td>Items will be moved to the transport.</td>
</tr>
<tr>
<td>WFS_CIM_RA_STACKER</td>
<td>Items will be moved to the intermediate stacker area.</td>
</tr>
<tr>
<td>WFS_CIM_RA_BILLCASSETTES</td>
<td>Items will be moved to item cassettes, i.e. cash-in and recycle cash units.</td>
</tr>
</tbody>
</table>

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSNULL</td>
<td>Take the default configuration.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Move items to the left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Move items to the right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Move items to the center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Move items to the top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Move items to the bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Move items to the front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Move items to the rear output position.</td>
</tr>
</tbody>
</table>

Output Param
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_CASHUNITERROR</td>
<td>A cash unit caused an error. A WFS_EXEE_CIM_CASHUNITERROR event will be sent with the details.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_UNSUPPOSITION</td>
<td>The position specified is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_INVALIDCASHUNIT</td>
<td>The cash unit number specified is not valid.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_FOREIGN_ITEMS_DETECTED</td>
<td>Foreign items have been detected in the input position.</td>
</tr>
</tbody>
</table>
## Events

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CIM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has been reached in one of the cash units.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>A cash unit caused an error.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM.MEDIADETECTED</td>
<td>Media was detected during the reset.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INPUT_P6</td>
<td>Level 2 and / or level 3 notes are detected during this operation.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>The items have been removed by the user.</td>
</tr>
<tr>
<td></td>
<td>This event is only generated if the bItemsTakenSensor field returned in</td>
</tr>
<tr>
<td></td>
<td>the Capabilities information is TRUE.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INFO_AVAILABLE</td>
<td>Information is available for items detected during the cash processing operation.</td>
</tr>
</tbody>
</table>

## Comments

None.
6.14 WFS_CMD_CIM_CONFIGURE_CASH_IN_UNITS

**Description**
This command is used to alter the banknote types a cash-in unit or recycle unit can take.
The values set by this command are persistent.

**Input Param**
LPWFSIMCASHINTYPE **lppCashInType**;

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

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

  - **usNumber**
    Logical number of the cash unit.

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

    | Value                      | Meaning                                                  |
    |----------------------------|----------------------------------------------------------|
    | WFS_CIM_CITYPALL           | The cash-in unit accepts all fit banknote types.         |
    | WFS_CIM_CITYPUNFIT         | The cash-in unit accepts all unfit banknotes.            |
    | WFS_CIM_CITYPINDIVIDUAL    | The cash-in unit or recycle unit accepts all types of fit banknotes specified in the following list. |
    | WFS_CIM_CITYPLEVEL2        | If a note handling standard is supported then level 2 note types are stored in this cash-in unit. |
    | WFS_CIM_CITYPLEVEL3        | If a note handling standard is supported then level 3 note types are stored in this cash-in unit. |
    | WFS_CIM_CITYPIPM           | The cash-in unit can accept items on the IPM interface.  |

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

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

**Output Param**
None.

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_SRVE_CIM_CASHUNITINFOCHANGED</td>
<td>A cash unit was changed.</td>
</tr>
</tbody>
</table>
Comments

None.
6.15 WFS_CMD_CIM_CONFIGURE_NOTETYPES

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

The values set by this command are persistent.

**Input Param**
LPUSHORT lpusNoteIDs;

*lpusNoteIDs*
Pointer to a zero-terminated list of unsigned shorts which contains the note IDs of the banknotes the banknote reader can accept.

**Output Param**
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_CASHINACTIVE</td>
<td>A cash-in transaction is active. This device requires that no cash-in transaction is active in order to perform the command.</td>
</tr>
</tbody>
</table>

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

**Comments**
None.
6.16 WFS_CMD_CIM_CREATE_P6_SIGNATURE

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

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

The bShutterControl field of the WFSCIMCAPS structure returned from the WFS_INF_CIM_CAPABILITIES query will determine whether the shutter is controlled implicitly by this command or whether the application must explicitly control the shutter using the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands, or WFS_CMD_CIM_PRESENT_MEDIA command. If bShutterControl is FALSE then this command does not operate the shutter in any way, the application is responsible for all shutter control. If bShutterControl is TRUE then this command opens and closes the shutter at various times during the command execution and the shutter is finally closed when all items are removed.

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

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

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

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

Input Param
None.

Output Param
LPWFSCIMP6SIGNATURE lpP6Signature;

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

usNoteId
Identification of note type.

ulLength
Length of the signature in bytes.

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

*lpSignature*
Pointer to the returned signature.

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

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_EXEE_CIM_INPUTREFUSE</td>
<td>The inserted item was no banknote or the note was not recognized.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSINSERTED</td>
<td>Items have been inserted into the cash-in position by the user.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>Items returned to the user have been taken.</td>
</tr>
<tr>
<td>Code</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>------------------------------------------------------------------</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSPRESENTED</td>
<td>Items have been presented to the user to be taken.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_NOTEERROR</td>
<td>An item detection error occurred.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INSERTITEMS</td>
<td>Device is ready to accept items from the user.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INFO_AVAILABLE</td>
<td>Information is available for items detected during this operation.</td>
</tr>
</tbody>
</table>

**Comments**  
None.
6.17 WFS_CMD_CIM_SET_GUIDANCE_LIGHT

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

Input Param
LPWFSCIMSETGUIDLIGHT lpSetGuidLight;
typedef struct _wfs_cim_set_guidlight {
    WORD     wGuidLight;
    DWORD     dwCommand;
} WFSCIMSETGUIDLIGHT, *LPWFSCIMSETGUIDLIGHT;

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_GUIDANCE_OFF</td>
<td>The light indicator turned off.</td>
<td>A</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_SLOW_FLASH</td>
<td>The light indicator is set to flash slowly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_MEDIUM_FLASH</td>
<td>The light indicator is set to flash medium frequency.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_QUICK_FLASH</td>
<td>The light indicator is set to flash quickly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_CONTINUOUS</td>
<td>The light indicator is turned on continuously (steady).</td>
<td>B</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_RED</td>
<td>The light indicator color is set to red.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_GREEN</td>
<td>The light indicator color is set to green.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_YELLOW</td>
<td>The light indicator color is set to yellow.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_BLUE</td>
<td>The light indicator color is set to blue.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_CYAN</td>
<td>The light indicator color is set to cyan.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_MAGENTA</td>
<td>The light indicator color is set to magenta.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_CIM_GUIDANCE_WHITE</td>
<td>The light indicator color is set to white.</td>
<td>C</td>
</tr>
</tbody>
</table>

Output Param
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_INVALID_PORT</td>
<td>An attempt to set a guidance light to a new value was invalid because the guidance light does not exist.</td>
</tr>
</tbody>
</table>

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

Comments
Guidance light support was added into the CIM primarily to support guidance lights for workstations where more than one instance of a CIM is present. The original SIU guidance light mechanism was not able to manage guidance lights for workstations with multiple CIMs. This command can also be used to set the status of the CIM guidance lights when only one instance of a CIM is present.
The slow and medium flash rates must not be greater than 2.0 Hz. It should be noted that in order to comply with American Disabilities Act guidelines only a slow or medium flash rate must be used.
6.18 WFS_CMD_CIM_CONFIGURE_NOTE_READER

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

Input Param
LPWFSCIMCONFIGURENOTEREADER lpConfigureNoteReader;

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

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

Output Param
LPWFSCIMCONFIGURENOTEREADEROUT lpConfigureNoteReaderOut;

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

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

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

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

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

Comments
None.
6.19 WFS_CMD_CIM_COMPARE_P6_SIGNATURE

**Description**

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

The reference signatures are created by the WFS_CMD_CIM_CREATE_P6_SIGNATURE command.

The transaction signatures are obtained through the WFS_INF_CIM_GET_P6_SIGNATURE command.

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

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

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

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

**Input Param**

LPWFSICMP6COMPARESIGNATURE lpP6CompareSignature;

typedef struct _wfs_cim_P6_compare_signature
{
  LPWFSICMP6SIGNATURE  *lppP6ReferenceSignatures;
  LPWFSICMP6SIGNATURE  *lppP6Signatures;
} WFSICMP6COMPARESIGNATURE, *LPWFSICMP6COMPARESIGNATURE;

*lppP6ReferenceSignatures*

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

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

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

*lppP6Signatures*

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

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

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

**Output Param**

LPWFSICMP6COMPARERESULT lpP6CompareResult;

typedef struct _wfs_cim_P6_compare_result
{
  USHORT    usCount;
  LPWFSICMP6SIGNATURESINDEX *lppP6SignaturesIndex;
} WFSICMP6COMPARERESULT, *LPWFSICMP6COMPARERESULT;

*usCount*

Number of WFSCIMP6SIGNATURESINDEX structures returned in *lppP6SignaturesIndex.*

*lppP6SignaturesIndex*

Pointer to a NULL-terminated array of pointers to WFSCIMP6SIGNATURESINDEX structures. This pointer is NULL and *usCount* is zero when the compare operation completes with no match found.
If there are matches found, `lppP6SignaturesIndex` contains the indexes of the matching signatures from the input parameter `lppP6Signatures`.

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

```c
typedef struct _wfs_cim_P6_signatures_index
{
    USHORT usIndex;
    USHORT usConfidenceLevel;
    ULONG ulLength;
    LPVOID lpComparisonData;
} WFS_CIMP6SIGNATURESINDEX, *LPWFS_CIMP6SIGNATURESINDEX;
```

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

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

- **ulLength**
  Length of the comparison data in bytes.

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

### Error Codes

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_CASHINACTIVE</td>
<td>A cash-in transaction is active.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in the exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_INVALIDREFSIG</td>
<td>At least one of the reference signatures is invalid. The application should prompt the operator to carefully retry the creation of the reference signatures.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_INVALIDTRNSIG</td>
<td>At least one of the transaction signatures is invalid.</td>
</tr>
</tbody>
</table>

### Events

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

### Comments

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

Description
This command activates or deactivates the power saving mode.

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

Input Param

\[
\text{LPWFSCIMPOWERSAVECONTROL lpPowerSaveControl;}
\]

\[
\text{typedef struct \_wfs\_cim\_power\_save\_control}
\]

\[
\begin{align*}
\text{USHORT } & \text{ usMaxPowerSaveRecoveryTime;}
\end{align*}
\]

\[
\text{WFSCIMPOWERSAVECONTROL, } \*\text{LPWFSCIMPOWERSAVECONTROL;}
\]

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

Output Param
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_POWERSAVETOOSHORT</td>
<td>The power saving mode has not been activated because the device is not able to resume from the power saving mode within the specified (\text{usMaxPowerSaveRecoveryTime}) value.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_POWERSAVEMEDIAPRESENT</td>
<td>The power saving mode has not been activated because media is present inside the device.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in an exchange state.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_SRVE_CIM_POWER_SAVE_CHANGE</td>
<td>The power save recovery time has changed.</td>
</tr>
</tbody>
</table>

Comments
None.
6.21 WFS_CMD_CIM_REPLENISH

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

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

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

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

Input Param
LPWFSCIMREP lpReplenish;

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

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

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

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

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

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

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

Output Param
LPWFSCIMREPRES lpReplenishResult;
typedef struct _wfs_cim_replenish_result
{
    ULONG    ulNumberOfItemsRemoved;
    ULONG    ulNumberOfItemsRejected;
    LPWFSCIMREPTARGETRES  *lppReplenishTargetResults;
} WFSCIMREPRES, *LPWFSCIMREPRES;

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

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

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

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

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

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

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

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

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

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

None.
6.22 WFS_CMD_CIM_SET_CASH_IN_LIMIT

Description

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

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

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

Input Param

LPWFSCIMCASHINLIMIT lpCashInLimit;

Pointer to the WFSCIMCASHINLIMIT structure. This cash-in limit structure can be used to limit the items that can be accepted during the cash-in operation. The limit set does not include counterfeit or suspected counterfeit items which may be detected during such a cash-in operation.

If the lpCashInLimit field is set to a NULL pointer there is no specific amount/number of items limit for the next cash-in operation. Note that the cash-in limit set by this command may itself be limited by the physical cash-in limitation of the device.

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

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

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

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

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

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

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

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

ulAmount
If set to a non-zero value, specifies a limit on the total amount of the cash-in operation. This value is expressed in minimum dispense units (see section WFS_INF_CIM_CURRENCY_EXP). If set to a zero value, this limitation will not be performed.
Output Param  None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in an exchange state.</td>
</tr>
</tbody>
</table>

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

Comments  None.
6.23 WFS_CMD_CIM_CASH_UNIT_COUNT

Description

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

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

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

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

Input Param

LPWFSCIMCOUNT lpCount;

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

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

usCount

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

lpusCUNumList

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

Output Param

None.

Error Codes

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

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

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

Events

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

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

Comments

None.
### 6.24 WFS_CMD_CIM_DEVICE_LOCK_CONTROL

**Description**

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

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

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

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

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

**Input Param**

LPWFSIMDEVICELOCKCONTROL lpDeviceLockControl;

```c
typedef struct _wfs_cim_device_lock_control
{
    WORD     wDeviceAction;
    WORD     wCashUnitAction;
    LPWFSIMUNITLOCKCONTROL *lppUnitLockControl;
} WFSIMDEVICELOCKCONTROL, *LPWFSIMDEVICELOCKCONTROL;
```

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_LOCK</td>
<td>Locks the CIM device so that it cannot be removed from its normal operating position.</td>
</tr>
<tr>
<td>WFS_CIM_UNLOCK</td>
<td>Unlocks the CIM device so that it can be removed from its normal operating position.</td>
</tr>
<tr>
<td>WFS_CIM_NOLOCKACTION</td>
<td>No lock/unlock action will be performed on the CIM device.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_LOCKALL</td>
<td>Locks all physical cash units supported.</td>
</tr>
<tr>
<td>WFS_CIM_UNLOCKALL</td>
<td>Unlocks all physical cash units supported.</td>
</tr>
<tr>
<td>WFS_CIM_LOCKINDIVIDUAL</td>
<td>Locks/unlocks physical cash units individually as specified in the <code>lppUnitLockControl</code> parameter.</td>
</tr>
<tr>
<td>WFS_CIM_NOLOCKACTION</td>
<td>No lock/unlock action will be performed on cash units.</td>
</tr>
</tbody>
</table>

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

```c
typedef struct _wfs_cim_unit_lock_control
{
    LPSTR     lPhysicalPositionName;
    WORD     wUnitAction;
} WFSIMUNITLOCKCONTROL, *LPWFSIMUNITLOCKCONTROL;
```
**lpPhysicalPositionName**

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

**wUnitAction**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_LOCK</td>
<td>Locks the specified cash unit so that it cannot be removed from the CIM device.</td>
</tr>
<tr>
<td>WFS_CIM_UNLOCK</td>
<td>Unlocks the specified cash unit so that it can be removed from the CIM device.</td>
</tr>
</tbody>
</table>

**Output Param**

None.

**Error Codes**

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

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

**Events**

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CIM_CASHUNITTHRESHOLD</td>
<td>A threshold condition has occurred in one of the cash units.</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>A problem occurred with a cash unit.</td>
</tr>
</tbody>
</table>

**Comments**

The normal command sequence is as follows:

Step1: WFS_CMD_CIM DEVICE LOCK CONTROL command is executed to unlock the device and some or all of the cash units.

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

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

Step 4: WFS_CMD_CIM DEVICE LOCK_CONTROL command is executed to lock the device and the cash units.

The relation of lock/unlock control with the WFS_CMD_CIM_START EXCHANGE and the WFS_CMD_CIM_END EXCHANGE commands is as follows:

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

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

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

Step 4: WFS_CMD_CIM_START EXCHANGE command is executed.
Step 5: WFS_CMD_CIM_END_EXCHANGE command is executed. During this command execution the Service Provider implicitly locks the device and all previously unlocked cash units. The cash unit status of the previously removed cash unit will be reset.
6.25 WFS_CMD_CIM_SET_MODE

Description
This execute command is used to set the deposit mode for the device and is only applicable for Mixed Media processing. The deposit mode determines how the device will process non cash items that are inserted. The deposit mode applies to all subsequent transactions. The deposit mode is persistent and is unaffected by a device reset by WFS_CMD_CIM_RESET or reset on another interface. The command will fail with a WFS_ERR_INVALID_DATA error where an attempt is made to set a mode that is not supported.

Input Param
LPWFS_CIMSETMODE lpMode;

typedef struct _wfs_cim_setmode
{
WORD wMixedMode;
} WFS_CIMSETMODE, *LPWFS_CIMSETMODE;

wMixedMode
Specifies the Mixed Media mode of the device as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_MIXEDMEDIANOTACTIVE</td>
<td>Mixed Media transactions are deactivated.</td>
</tr>
<tr>
<td></td>
<td>This is the default mode.</td>
</tr>
<tr>
<td>WFS_CIM_IPMMIXEDMEDIA</td>
<td>Mixed Media transactions are activated in combination with the IPM interface as defined by the capability wMixedMode.</td>
</tr>
</tbody>
</table>

Output Param
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_CASHINACTIVE</td>
<td>A cash-in transaction is active.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_MEDIAINACTIVE</td>
<td>An item processing transaction is active.</td>
</tr>
</tbody>
</table>

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

Comments
The commands WFS_CMD_CIM_SET_MODE and WFS_CMD_IPM_SET_MODE are equivalent; an application can use either to control the Mixed Media mode. If the requested mode is already active WFS_CMD_CIM_SET_MODE command returns with WFS_SUCCESS.
6.26 WFS_CMD_CIM_PRESENT_MEDIA

Description
This command opens the shutter and presents items to be taken by the customer after a WFS_CMD_CIM_CASH_IN, WFS_CMD_CIM_ROLLBACK, WFS_CMD_CIM_RESET or WFS_CMD_CIM_CREATE_P6_SIGNATURE command. The command is only valid on positions where fwUsage reported by the WFS_INF_CIM_POSITION_CAPABILITIES command is WFS_CIM_POSROLLBACK or WFS_CIM_POSREFUSE and where bPresentControl reported by the WFS_INF_CIM_POSITION_CAPABILITIES command is FALSE.

This command cannot be used to present items stacked through the CDM interface. Where this is attempted the command fails with a WFS_ERR_SEQUENCE_ERROR error.

Mixed Media Mode: If the device is operating in Mixed Media mode (WFSCIMSTATUS.wMixedMode == WFS_CIM_IPMMIXEDMEDIA) this command will not perform any operation unless the WFS_CMD_IPM_PRESENT_MEDIA command is called or has already been called on the IPM interface. Shutter control on devices that support Mixed Media processing is always implicit.

Input Param
LPWFSCIMPRESSENT lpPresent;

If the input parameter is NULL then all refused items are returned from all positions in a sequence determined by the Service Provider.

typedef struct _wfs_cim_present
{
    WORD     fwPosition;
} WFSCIMPRESENT, *LPWFSCIMPRESENT;

fwPosition
Describes the position where the media is to be presented as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSNULL</td>
<td>The default configuration information should be used.</td>
</tr>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>Present items to the left input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>Present items to the right input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>Present items to the center input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>Present items to the top input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>Present items to the bottom input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRONT</td>
<td>Present items to the front input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>Present items to the rear input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Present items to the left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Present items to the right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Present items to the center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Present items to the top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Present items to the bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Present items to the front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Present items to the rear output position.</td>
</tr>
</tbody>
</table>

Output Param
None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_CIM_UNSUPPOSITION</td>
<td>The position specified is not supported or is not a valid position for this command.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_SHUTTERNOTOPEN</td>
<td>Shutter failed to open.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_NOITEMS</td>
<td>There were no items to present at the specified position.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_EXCHANGEACTIVE</td>
<td>The CIM is in an exchange state.</td>
</tr>
<tr>
<td>WFS_ERR_CIM_FOREIGN_ITEMS_DETECTED</td>
<td>Foreign items have been detected in the input position.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>The items have been removed by the user. This event is only generated if the \textit{bItemsTakenSensor} field returned in the capabilities information is TRUE.</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSPRESENTED</td>
<td>Items have been presented to the user to be taken.</td>
</tr>
</tbody>
</table>

**Comments** None.
### 7. Events

#### 7.1 WFS_SRVE_CIM_SAFEDOOROPEN

<table>
<thead>
<tr>
<th>Description</th>
<th>This service event specifies that the safe door has been opened.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Param</td>
<td>None.</td>
</tr>
<tr>
<td>Comments</td>
<td>None.</td>
</tr>
</tbody>
</table>
### 7.2 WFS_SRVE_CIM_SAFEDOORCLOSED

<table>
<thead>
<tr>
<th>Description</th>
<th>This service event specifies that the safe door has been closed.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Param</td>
<td>None.</td>
</tr>
<tr>
<td>Comments</td>
<td>None.</td>
</tr>
</tbody>
</table>
7.3 **WFS_USRE_CIM_CASHUNITTHRESHOLD**

**Description**
This user event specifies that a threshold condition has occurred in one of the cash units or the threshold condition is removed. If the cash unit is a shared cash unit in a compound device then this event can also be generated as a result of an operation on another device class.

**Event Param**
LPWFSCIMCASHIN lpCashUnit;

*lpCashUnit*
Pointer to a WFSCIMCASHIN structure, describing the cash unit on which the threshold condition occurred. See *lpCashUnit->usStatus* for the type of condition. For a description of the WFSCIMCASHIN structure, see the definition of the WFS_INF_CIM_CASH_UNIT_INFO command.

**Comments**
None.
7.4 **WFS_SRVE_CIM_CASHUNITINFOCHANGED**

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

- WFS_CMD_CIM_SET_CASH_UNIT_INFO
- WFS_CMD_CIM_END_EXCHANGE

If the cash unit is a shared cash unit in a compound device then this event can also be generated as a result of an operation on another device class.

**Event Param**

- LPWFSCIMCASHIN lpCashUnit;

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

**Comments**
None.
7.5 WFS_SRVE_CIM_TELLERINFOCHANGED

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

**Event Param**  
LPUSHORT lpusTellerID;

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

**Comments**  
None.
7.6 WFS_EXEE_CIM_CASHUNITERROR

Description  This execute event specifies that a cash unit was addressed which caused a problem.

Event Param  LPWFSCIMCUERROR lpCashUnitError;

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

wFailure  Specifies the kind of failure that occurred in the cash unit. Values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_CASHUNITEMPTY</td>
<td>Specified cash unit is empty.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITERROR</td>
<td>Specified cash unit has malfunctioned.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITFULL</td>
<td>Specified cash unit is full.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITLOCKED</td>
<td>The bAppLock field of the WFSCIMCASHIN structure has previously</td>
</tr>
<tr>
<td></td>
<td>been set to TRUE and the cash unit remains locked.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITNOTCONF</td>
<td>Specified cash unit is not configured due to being removed and/or</td>
</tr>
<tr>
<td></td>
<td>replaced with a different cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITINVALID</td>
<td>Specified cash unit is invalid.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITCONFIG</td>
<td>Attempt to change the setting of a self-configuring cash unit.</td>
</tr>
<tr>
<td>WFS_CIM_FEEDMODULEPROBLEM</td>
<td>A problem has been detected with the feeding module.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITPHYSICALLOCKED</td>
<td>The cash unit could not be unlocked by the WFS_CMD_CIM_DEVICE_LOCK_-</td>
</tr>
<tr>
<td></td>
<td>CONTROL command and remains physically locked.</td>
</tr>
<tr>
<td>WFS_CIM_CASHUNITPHYSICALUNLOCKED</td>
<td>The cash unit could not be locked by the WFS_CMD_CIM_DEVICE_LOCK_-</td>
</tr>
<tr>
<td></td>
<td>CONTROL command and remains physically unlocked.</td>
</tr>
</tbody>
</table>

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

Comments  None.
### 7.7 WFS_SRVE_CIM_ITEMSTAKEN

**Description**
This service event specifies that items presented to the user have been taken. This event may be generated at any time.

**Event Param**
`LPWFSCIMPOSITIONINFO lpPositionInfo;`

```
typedef struct _wfs_cim_position_info
{
    WORD     wPosition;
    WORD     wAdditionalBunches;
    USHORT   usBunchesRemaining;
} WFSCIMPOSITIONINFO, *LPWFSCIMPOSITIONINFO;
```

**wPosition**
Specifies the position from which the items have been taken, set to one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>Items taken from the left input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>Items taken from the right input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>Items taken from the center input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>Items taken from the top input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>Items taken from the bottom input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRACT</td>
<td>Items taken from the front input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>Items taken from the rear input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Items taken from the left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Items taken from the right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Items taken from the center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Items taken from the top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Items taken from the bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Items taken from the front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Items taken from the rear output position.</td>
</tr>
</tbody>
</table>

**wAdditionalBunches**
This value will always be zero within this event.

**usBunchesRemaining**
This value will always be zero within this event.

**Comments**
None.
7.8  WFS_SRVE_CIM_COUNTS_CHANGED

Description  This service event is generated if the device is a compound device and the counts in a shared cash unit have changed as a result of an operation on the other device class other than as a result of an operation that explicitly sets counts. For example, WFS_CMD_CDM_SET_CASH_UNIT_INFO and WFS_CMD_CDM_END_EXCHANGE commands on the CDM and WFS_CMD_IPM_SET_MEDIA_BIN_INFO command on the IPM.

Event Param  LPWFS_CIMCOUNTSCALLED lpCountsChanged;

typedef struct _wfs_cim_counts_changed
{
    USHORT    usCount;
    LPUSHORT    lpusCUNumList;
} WFS_CIMCOUNTSCALLED, *LPWFS_CIMCOUNTSCALLED;

usCount
The size of lpusCUNumList.

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

Comments  None.
7.9 WFS_EXEE_CIM_INPUTREFUSE

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

Event Param
LPUSHORT lpusReason;

lpusReason
Pointer to an USHORT holding the reason for refusing a part of the amount. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_CASHINUNITFULL</td>
<td>Cash unit is full.</td>
</tr>
<tr>
<td>WFS_CIM_INVALIDBILL</td>
<td>Recognition of the items took place, but one or more of the items are invalid.</td>
</tr>
<tr>
<td>WFS_CIM_NOBILLSTODEPOSIT</td>
<td>A deposit has failed for a reason not covered by the other reasons and the failure is not a fatal hardware problem.</td>
</tr>
<tr>
<td>WFS_CIM_DEPOSITFAILURE</td>
<td>Failure of a common input component which is shared by all cash units.</td>
</tr>
<tr>
<td>WFS_CIM_COMMINPCOMPFAILURE</td>
<td>The intermediate stacker is full.</td>
</tr>
<tr>
<td>WFS_CIM_STACKERFULL</td>
<td>Foreign items have been detected in the input position.</td>
</tr>
<tr>
<td>WFS_CIM_FOREIGN_ITEMS_DETECTED</td>
<td>Recognition of the items did not take place. The bunch of notes presented is invalid, e.g. it is too large or was presented incorrectly.</td>
</tr>
<tr>
<td>WFS_CIM_INVALIDBUNCH</td>
<td>One or more counterfeit items have been detected and refused. This is only applicable to devices which do not support a legislative note handling standard and are capable of differentiating between invalid and counterfeit items.</td>
</tr>
<tr>
<td>WFS_CIM_COUNTERFEIT</td>
<td>Number of items count exceeded the limitation set with the WFS_CMD_CIM_SET_CASH_IN_LIMIT command.</td>
</tr>
<tr>
<td>WFS_CIM_LIMITOVERTOTAITEMS</td>
<td>Amount exceeded the limitation set with the WFS_CMD_CIM_SET_CASH_IN_LIMIT command.</td>
</tr>
</tbody>
</table>

Comments
None.
7.10 WFS_SRVE_CIM_ITEMSPRESENTED

Description
This service event specifies that items have been presented to the output position, and the shutter
has been opened to allow the user to take the items.

Event Param
LPWFSCIMPOSITIONINFO lpPositionInfo;

typedef struct _wfs_cim_position_info
{
    WORD     wPosition;
    WORD     wAdditionalBunches;
    USHORT   usBunchesRemaining;
} WFSCIMPOSITIONINFO, *LPWFSCIMPOSITIONINFO;

wPosition
Specifies the position from which the items have been presented, set to one of the following
values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Items presented at the left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Items presented at the right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Items presented at the center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Items presented at the top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Items presented at the bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Items presented at the front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Items presented at the rear output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>Items presented at the left input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>Items presented at the right input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>Items presented at the center input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>Items presented at the top input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>Items presented at the bottom input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRONT</td>
<td>Items presented at the front input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>Items presented at the rear input position.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_ADDBUNCHNONE</td>
<td>No additional bunches remain.</td>
</tr>
<tr>
<td>WFS_CIM_ADDBUNCHONEMORE</td>
<td>At least one additional bunch remains.</td>
</tr>
<tr>
<td>WFS_CIM_ADDBUNCHUNKNOWN</td>
<td>It is unknown whether additional bunches remain.</td>
</tr>
</tbody>
</table>

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

Comments
None.
## 7.11 WFS_SRVE_CIM_ITEMSINSERTED

### Description
This service event specifies that items have been inserted into the cash-in position by the user. This event may be generated at any time.

### Event Param
`LPWFSCIMPOSITIONINFO lpPositionInfo;`

```c
typedef struct _wfs_cim_position_info
{
    WORD     wPosition;
    WORD     wAdditionalBunches;
    USHORT   usBunchesRemaining;
}WFSCIMPOSITIONINFO, *LPWFSCIMPOSITIONINFO;
```

**wPosition**
Specifies the position where the items have been inserted, set to one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_POSINLEFT</td>
<td>Items detected in the left input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINRIGHT</td>
<td>Items detected in the right input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINCENTER</td>
<td>Items detected in the center input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINTOP</td>
<td>Items detected in the top input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINBOTTOM</td>
<td>Items detected in the bottom input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINFRONT</td>
<td>Items detected in the front input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSINREAR</td>
<td>Items detected in the rear input position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTLEFT</td>
<td>Items detected in the left output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTRIGHT</td>
<td>Items detected in the right output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTCENTER</td>
<td>Items detected in the center output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTTOP</td>
<td>Items detected in the top output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTBOTTOM</td>
<td>Items detected in the bottom output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTFRONT</td>
<td>Items detected in the front output position.</td>
</tr>
<tr>
<td>WFS_CIM_POSOUTREAR</td>
<td>Items detected in the rear output position.</td>
</tr>
</tbody>
</table>

**wAdditionalBunches**
This value will always be zero within this event.

**usBunchesRemaining**
This value will always be zero within this event.

### Comments
None.
### 7.12 WFS_EXEE_CIM_NOTEERROR

**Description**
This execute event specifies the reason for an item detection error during an operation which involves moving items.

**Event Param**
LPUSHORT lpusReason;

*lpusReason*
Pointer to an USHORT holding the reason for the item detection error. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_DOUBLENOTEDETECTED</td>
<td>Double notes have been detected.</td>
</tr>
<tr>
<td>WFS_CIM_LONGNOTEDETECTED</td>
<td>A long note has been detected.</td>
</tr>
<tr>
<td>WFS_CIM_SKEWEDNOTE</td>
<td>A skewed note has been detected.</td>
</tr>
<tr>
<td>WFS_CIM_INCORRECTCOUNT</td>
<td>An item counting error has occurred.</td>
</tr>
<tr>
<td>WFS_CIM_NOTESTOOCLOSE</td>
<td>Notes have been detected as being too close.</td>
</tr>
<tr>
<td>WFS_CIM_OTHERNOTEERROR</td>
<td>An item error not covered by the other values has been detected.</td>
</tr>
<tr>
<td>WFS_CIM_SHORTNOTEDETECTED</td>
<td>A short note has been detected.</td>
</tr>
</tbody>
</table>

**Comments**
None.
### 7.13 WFS_EXEE_CIM_SUBCASHIN

<table>
<thead>
<tr>
<th>Description</th>
<th>This execute event is generated when one of the sub cash-in operations into which the cash-in operation was divided has finished successfully.</th>
</tr>
</thead>
</table>
| Event Param | LPWFSCIMNOTENUMBERLIST lpNoteNumberList;  
  
  `lpNoteNumberList`  
  Pointer to a WFSCIMNOTENUMBERLIST structure holding a list of banknote numbers which have been identified and accepted during execution of the sub cash-in. This field will contain the banknote numbers of the accepted items. For a description of the WFSCIMNOTENUMBERLIST structure see the definition of the WFS_INF_CIM_CASH_UNIT_INFO command. |
| Comments    | None.                                                                                                           |
7.14 WFS_SRVE_CIM_MEDIADETECTED

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

Event Param
LPWFSCIMITEMPOSITION lpItemPosition;
For a description of this parameter see the definition of the WFS_CMD_CIM_RESET command.

Comments
None.
7.15 WFS_EXEE_CIM_INPUT_P6

Description
This execute event is generated if level 2 and / or level 3 notes are detected during the cash processing operation.

Event Param
LPWFSCIMP6INFO *lppP6Info;

Pointer to a NULL-terminated array of pointers to WFSCIMP6INFO structures, one structure for every level. For the description of the structure see the definition of the WFS_INF_CIM_GET_P6_INFO command.

Comments
None.
7.16 WFS_EXEE_CIM_INFO_AVAILABLE

**Description**  
This execute event is generated when information is available for items detected during the cash processing operation.

**Event Param**  
LPWFSCIMITEMINFOSUMMARY *lppItemInfoSummary;

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

```c
typedef struct _wfs_cim_item_info_summary
{
    USHORT usLevel;
    USHORT usNumOfItems;
} WFSCIMITEMINFOSUMMARY, *LPWFSCIMITEMINFOSUMMARY;
```

**usLevel**  
Defines the note level. Possible values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_LEVEL_2</td>
<td>Information for level 2 notes.</td>
</tr>
<tr>
<td>WFS_CIM_LEVEL_3</td>
<td>Information for level 3 notes.</td>
</tr>
<tr>
<td>WFS_CIM_LEVEL_4</td>
<td>Information for level 4 notes.</td>
</tr>
</tbody>
</table>

**usNumOfItems**  
Number of items classified as usLevel which have information available.

**Comments**  
None.
7.17 WFS_EXEE_CIM_INSERTITEMS

<table>
<thead>
<tr>
<th>Description</th>
<th>This event notifies the application when the device is ready for the user to insert items.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Param</td>
<td>None.</td>
</tr>
<tr>
<td>Comments</td>
<td>None.</td>
</tr>
</tbody>
</table>
7.18 WFS_SRVE_CIM_DEVICEPOSITION

Description
This service event reports that the device has changed its position status.

Event Param
```
typedef struct _wfs_cim_device_position
{
    WORD     wPosition;
} WFS_CIMDEVICEPOSITION, *LPWFS_CIMDEVICEPOSITION;
```

```
wPosition
Position of the device as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CIM_DEVICEINPOSITION</td>
<td>The device is in its normal operating position.</td>
</tr>
<tr>
<td>WFS_CIM_DEVICENOTINPOSITION</td>
<td>The device has been removed from its normal operating position.</td>
</tr>
<tr>
<td>WFS_CIM_DEVICEPOSUNKNOWN</td>
<td>The position of the device cannot be determined.</td>
</tr>
</tbody>
</table>
```

Comments
None.
7.19 WFS_SRVE_CIMPOWER_SAVE_CHANGE

**Description**
This service event specifies that the power save recovery time has changed.

**Event Param**
LPWFSCIMPOWERSAVECHANGE lpPowerSaveChange;

typedef struct _wfs_cim_power_save_change
{
    USHORT    usPowerSaveRecoveryTime;
} WFSCIMPOWERSAVECHANGE, *LPWFSCIMPOWERSAVECHANGE;

*usPowerSaveRecoveryTime*
Specifies the actual number of seconds required by the device to resume its normal operational state. This value is zero if the device exited the power saving mode.

**Comments**
If another device class compounded with this device enters into a power saving mode, this device will automatically enter into the same power saving mode and this event will be generated.
7.20 WFS_EXEE_CIM_INCOMPLETEREPLENISH

**Description**

This execute event is generated when some items had been moved before the WFS_CMD_CIM_REPLENISH command failed with an error code (not WFS_SUCCESS), but some items were moved then the details will be reported with this event. This event can only occur once per command.

**Event Param**

LPWFS CIM INCOMPLETEREPLENISH lpIncompleteReplenish;

typedef struct _wfs_cim_incomplete_replenish
{
    LPWFS CIM REPRES lpReplenish;
} WFS CIM INCOMPLETEREPLENISH, *LPWFS CIM INCOMPLETEREPLENISH;

lpReplenish

The WFSCIMREPRES structure is defined in the description of the command WFS_CMD_CIM_REPLENISH. Note that in this case the values in this structure report the amount and number of each denomination that have actually been moved during the replenishment command.

**Comments**

None.
8. ATM Cash-In Transaction Flow - Application Guidelines

The following table is a summary of the application flows required given the possible values for \textit{bShutterControl} and \textit{bItemsTakenSensor} for a successful cash-in transaction. In all cases \textit{bPresentControl} == TRUE.

<table>
<thead>
<tr>
<th>\textbf{bShutterControl} == TRUE</th>
<th>\textbf{bItemsInsertedSensor} == TRUE</th>
<th>\textbf{bItemsInsertedSensor} == FALSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CMD_CIM_CASH_IN_START</td>
<td>WFS_CMD_CIM_CASH_IN_START</td>
<td>WFS_CMD_CIM_CASH_IN_START</td>
</tr>
<tr>
<td>WFS_CMD_CIM_CASH_IN</td>
<td>WFS_CMD_CIM_CASH_IN</td>
<td>WFS_CMD_CIM_CASH_IN</td>
</tr>
<tr>
<td>InsertedEvent generated</td>
<td>InsertedEvent generated</td>
<td>InsertedEvent generated</td>
</tr>
<tr>
<td>WFS_CMD_CIM_CASH_IN_END</td>
<td>WFS_CMD_CIM_CASH_IN_END</td>
<td>WFS_CMD_CIM_CASH_IN_END</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>\textbf{bShutterControl} == FALSE</th>
<th>\textbf{bItemsInsertedSensor} == TRUE</th>
<th>\textbf{bItemsInsertedSensor} == FALSE</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CMD_CIM_CASH_IN_START</td>
<td>WFS_CMD_CIM_CASH_IN_START</td>
<td>WFS_CMD_CIM_CASH_IN_START</td>
</tr>
<tr>
<td>WFS_CMD_CIM_OPEN_SHUTTER</td>
<td>WFS_CMD_CIM_OPEN_SHUTTER</td>
<td>WFS_CMD_CIM_OPEN_SHUTTER</td>
</tr>
<tr>
<td>InsertedEvent generated</td>
<td>InsertedEvent generated</td>
<td>InsertedEvent generated</td>
</tr>
<tr>
<td>WFS_CMD_CIM_CLOSE_SHUTTER</td>
<td>WFS_CMD_CIM_CLOSE_SHUTTER</td>
<td>WFS_CMD_CIM_CLOSE_SHUTTER</td>
</tr>
<tr>
<td>WFS_CMD_CIM_CASH_IN</td>
<td>WFS_CMD_CIM_CASH_IN</td>
<td>WFS_CMD_CIM_CASH_IN</td>
</tr>
<tr>
<td>WFS_CMD_CIM_CASH_IN_END</td>
<td>WFS_CMD_CIM_CASH_IN_END</td>
<td>WFS_CMD_CIM_CASH_IN_END</td>
</tr>
</tbody>
</table>

The following sections describe the flow of a cash-in transaction on a Self-Service CIM. These application flows are provided as guidelines only.
8.1 OK Transaction (Explicit Shutter Control)

The following table describes a normal cash-in transaction flow where everything works and the shutter is explicitly controlled by the application.

This flow covers the following cases:

- \( b\text{ShutterControl} = \) FALSE, \( b\text{ItemsInsertedSensor} = \) TRUE
- \( b\text{ShutterControl} = \) FALSE, \( b\text{ItemsInsertedSensor} = \) FALSE

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer</th>
<th>Application</th>
<th>XFS Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Customer selects cash-in operation.</td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN_START</td>
</tr>
<tr>
<td>2.</td>
<td>Open the shutter of the input tray.</td>
<td></td>
<td>WFS_CMD_CIM_OPEN_SHUTTER,WFS_SRV_CIM_ITEMSPRESENTED</td>
</tr>
<tr>
<td>3.</td>
<td>Ask the customer to insert money.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>If ( b\text{ItemsInsertedSensor} = ) FALSE, confirm completion.</td>
<td>If ( b\text{ItemsInsertedSensor} = ) TRUE: WFS_SRV_CIM_ITEMSINSERTED</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Close shutter.</td>
<td></td>
<td>WFS_CMD_CIM_CLOSE_SHUTTER</td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN completion of WFS_CMD_CIM_CASH_IN</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Display the number of bills and/or amount recognized so far.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Ask the customer for further actions:</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If he wants to insert more money: Repeat from step 2.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If he wants to finish the transaction: Continue with step 10.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If he wants to get back all items inserted so far see table &quot;Cancellation by Customer (Explicit Shutter Control)&quot;</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>Transport the money into the cash units RECYCLE_UNIT/CASHINBOX</td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN_END</td>
</tr>
<tr>
<td>11.</td>
<td>Credit the money to the customer's account.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>End of transaction.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.2 Cancellation by Customer (Explicit Shutter Control)

The following table describes the flow of a cash-in transaction where the customer wants all the items to be returned after recognition.

This flow covers the following cases:

- $b_{ShutterControl} = \text{FALSE, } b_{ItemsInsertedSensor} = \text{TRUE, } b_{ItemsTakenSensor} = \text{TRUE}$
- $b_{ShutterControl} = \text{FALSE, } b_{ItemsInsertedSensor} = \text{FALSE, } b_{ItemsTakenSensor} = \text{TRUE}$
- $b_{ShutterControl} = \text{FALSE, } b_{ItemsInsertedSensor} = \text{TRUE, } b_{ItemsTakenSensor} = \text{FALSE}$
- $b_{ShutterControl} = \text{FALSE, } b_{ItemsInsertedSensor} = \text{FALSE, } b_{ItemsTakenSensor} = \text{FALSE}$

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer</th>
<th>Application</th>
<th>XFS Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.-9.</td>
<td>See OK Transaction (Explicit Shutter Control).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>Selection: Return all the items.</td>
<td>Transport the items recognized to the output position.</td>
<td>WFS_CMD_CIM_CASH_IN_ROLLBACK</td>
</tr>
<tr>
<td>11.</td>
<td>Open shutter.</td>
<td></td>
<td>WFS_CMD_CIM_OPEN_SHUTTER WFS_SRVE_CIM_ITEMSPRESENTED</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Request removal of the money.</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>If $b_{ItemsTakenSensor} = \text{FALSE}$, confirm completion or use application timeout.</td>
<td>If $b_{ItemsTakenSensor} = \text{TRUE}$:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>Close shutter.</td>
<td></td>
<td>WFS_CMD_CIM_CLOSE_SHUTTER</td>
</tr>
</tbody>
</table>
8.3 Stacker Becomes Full (Explicit Shutter Control)

The following table describes the flow of a cash-in transaction when the stacker becomes full during the transaction and the shutter is explicitly controlled by the application. This flow covers the following cases:

- \( b_{\text{ShutterControl}} == \text{FALSE}, b_{\text{ItemsInsertedSensor}} == \text{TRUE}, b_{\text{ItemsTakenSensor}} == \text{TRUE} \)
- \( b_{\text{ShutterControl}} == \text{FALSE}, b_{\text{ItemsInsertedSensor}} == \text{FALSE}, b_{\text{ItemsTakenSensor}} == \text{TRUE} \)
- \( b_{\text{ShutterControl}} == \text{FALSE}, b_{\text{ItemsInsertedSensor}} == \text{TRUE}, b_{\text{ItemsTakenSensor}} == \text{FALSE} \)
- \( b_{\text{ShutterControl}} == \text{FALSE}, b_{\text{ItemsInsertedSensor}} == \text{FALSE}, b_{\text{ItemsTakenSensor}} == \text{FALSE} \)

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer</th>
<th>Application</th>
<th>XFS Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.-6.</td>
<td>See OK Transaction (Explicit Shutter Control).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WFS_EXEE_CIM_INPUTREFUSE (StackerFull) and</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN completes</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>with WFS_SUCCESS</td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td></td>
<td>WFS_CMD_CIM_OPEN_SHUTTER</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>WFS_SRVE_CIM_ITEMSPRESENTED</td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>If ( b_{\text{ItemsTakenSensor}} == \text{FALSE} ): confirm completion or use application timeout.</td>
<td>If ( b_{\text{ItemsTakenSensor}} == \text{TRUE} ):</td>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
</tr>
<tr>
<td>12.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td></td>
<td></td>
<td>WFS_CMD_CIM_CLOSE_SHUTTER</td>
</tr>
<tr>
<td>14.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>Transport the money into the cash units</td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN_END</td>
</tr>
<tr>
<td></td>
<td>RECYCLE_UNIT/CASHINBOX.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

122
8.4 Bill Recognition Error (Explicit Shutter Control)

The following table describes the flow of a cash-in transaction when the items are rejected as unrecognized during the transaction and the shutter is explicitly controlled by the application.

This flow covers the following cases:

- \( b_{\text{ShutterControl}} = \text{FALSE}, b_{\text{ItemsInsertedSensor}} = \text{TRUE}, b_{\text{ItemsTakenSensor}} = \text{TRUE} \)
- \( b_{\text{ShutterControl}} = \text{FALSE}, b_{\text{ItemsInsertedSensor}} = \text{FALSE}, b_{\text{ItemsTakenSensor}} = \text{TRUE} \)
- \( b_{\text{ShutterControl}} = \text{FALSE}, b_{\text{ItemsInsertedSensor}} = \text{TRUE}, b_{\text{ItemsTakenSensor}} = \text{FALSE} \)
- \( b_{\text{ShutterControl}} = \text{FALSE}, b_{\text{ItemsInsertedSensor}} = \text{FALSE}, b_{\text{ItemsTakenSensor}} = \text{FALSE} \)

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer</th>
<th>Application</th>
<th>XFS Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.-6.</td>
<td>See OK Transaction (Explicit Shutter Control).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN WFS_EXEE_CIM_INPUTREFUSE (InvalidBill) and completion of WFS_CMD_CIM_CASH_IN with WFS_SUCCESS</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td>Open shutter.</td>
<td>WFS_CMD_CIM_OPEN_SHUTTER WFS_SRVE_CIM_ITEMSPRESENTED</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Tell the customer that the bills were not recognized and that he should take the bills.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>Customer removes unrecognized money</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>If ( b_{\text{ItemsTakenSensor}} = \text{FALSE} ): confirm completion or use application timeout.</td>
<td>If ( b_{\text{ItemsTakenSensor}} = \text{TRUE} ): WFS_SRVE_CIM_ITEMSTAKEN</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>Close shutter.</td>
<td>WFS_CMD_CIM_CLOSE_SHUTTER</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>Display the amount recognized so far.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>Ask the customer for further actions:</td>
<td>If he wants to deposit the amount: Continue with step 15. If he wants to get back all items inserted so far see table &quot;Cancellation by Customer (Explicit Shutter Control)&quot;</td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>Transport the money into the cash units RECYCLE_UNIT/CASHINBOX.</td>
<td>WFS_CMD_CIM_CASH_IN_END</td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td>Credit the money to the customer's account.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.</td>
<td>End of transaction.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.5 OK Transaction (Implicit Shutter Control)

The following table describes a normal cash-in transaction flow where everything works and the shutter is implicitly controlled by the Service Provider. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are not explicitly used by the application.

This flow covers the following cases:
- $b\text{ShutterControl} = \text{TRUE}$, $b\text{ItemsInsertedSensor} = \text{TRUE}$
- $b\text{ShutterControl} = \text{TRUE}$, $b\text{ItemsInsertedSensor} = \text{FALSE}$

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer</th>
<th>Application</th>
<th>XFS Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Customer selects cash-in operation.</td>
<td>WFS_CMD_CIM_CASH_IN_START</td>
<td></td>
</tr>
</tbody>
</table>
| 2.   | | WFS_CMD_CIM_CASH_IN  
The Service Provider opens the shutter.  
WFS_EXEE_CIM_INSERTITEMS event is sent when the shutter is fully open and the device is ready to begin accepting items. |
| 3.   | | Ask the customer to insert money. |
| 4.   | Customer inserts money. | If $b\text{ItemsInsertedSensor} = \text{TRUE}$:  
WFS_SRVE_CIM_ITEMSINSERTED |
| 5.   | | The Service Provider closes the shutter and begins bill recognition.  
The WFS_CMD_CIM_CASH_IN command completes. |
| 7.   | | Display the number of bills and/or amount recognized so far. |
| 8.   | | Ask the customer for further actions:  
If he wants to insert more money:  
Repeat from step 2.  
If he wants to finish the transaction:  
Continue with step 9.  
If he wants to get back all items inserted so far see table  
"Cancellation by Customer (Implicit Shutter Control)" |
| 9.   | | Transport the money into the cash units  
RECYCLE_UNIT/CASHINBOX.  
WFS_CMD_CIM_CASH_IN_END |
| 10.  | | Credit the money to the customer’s account. |
| 11.  | | End of transaction. |
8.6 Cancellation by Customer (Implicit Shutter Control)

The following table describes the flow of a cash-in transaction where the customer wants all the items to be returned after recognition and the shutter is implicitly controlled by the Service Provider. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are not used.

This flow covers the following cases:

- \( bShutterControl == \text{TRUE}, bItemsInsertedSensor == \text{TRUE}, bItemsTakenSensor == \text{TRUE} \)
- \( bShutterControl == \text{TRUE}, bItemsInsertedSensor == \text{TRUE}, bItemsTakenSensor == \text{FALSE} \)

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer</th>
<th>Application</th>
<th>XFS Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-8.</td>
<td>See OK Transaction.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td>Selection: Return all the items.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| 10. | Transport the items recognized to the output position. | WFS_CMD_CIM_CASH_IN_ROLLBACK
WFS_SRV_CIM_ITEMSPRESENTED |
| 11. | Request removal of the money. | | |
| 12. | Customer takes the money from the output position. | | |
| 13. | If \( bItemsTakenSensor == \text{FALSE} \): confirm completion or use application timeout. | If \( bItemsTakenSensor == \text{TRUE} \):
WFS_SRV_CIM_ITEMSTAKEN
The Service Provider closes the shutter. |
| 14. | End of transaction | |
8.7 Implicit Control of the Shutter - WFS_EXEE_CIM_SUBCASHIN event

The following table describes the chronological steps taken in the flow of a cash-in transaction where the cash-in operation is subdivided into a number of logical operations under hardware control. In this case a WFS_EXEE_CIM_SUBCASHIN event is generated for each sub cash-in operation. This may be the case for instance where a device does its coin or bill recognition in batches of 25. In this case the Service Provider would post a WFS_EXEE_CIM_SUBCASHIN event each time 25 coins were processed. In this example the shutter is implicitly controlled by the Service Provider so the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are not used.

This flow covers the following cases:

- \( b_{\text{ShutterControl}} = \text{TRUE}, b_{\text{ItemsInsertedSensor}} = \text{TRUE} \)
- \( b_{\text{ShutterControl}} = \text{TRUE}, b_{\text{ItemsInsertedSensor}} = \text{FALSE} \)

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer Application</th>
<th>XFS Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.-5.</td>
<td>See OK Transaction (Implicit Shutter Control).</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td>The Service Provider closes the shutter and begins bill recognition.</td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td>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.</td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td>The WFS_CMD_CIM_CASH_IN command completes.</td>
</tr>
<tr>
<td>9.</td>
<td>Display the number of bills and/or amount recognized so far.</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>Ask the customer for further actions: If he wants to insert more money: Repeat from step 2. If he wants to finish the transaction: Continue with step 11. If he wants to get back all items inserted so far see table &quot;Cancellation by Customer (Implicit Shutter Control)&quot;</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN_END</td>
</tr>
<tr>
<td>12.</td>
<td></td>
<td>End of transaction.</td>
</tr>
</tbody>
</table>
8.8 **OK Transaction - Note Handling Standard Supported**

This section describes a possible cash-in transaction where a note handing standard is supported and everything works fine when level 2 / level 3 notes are inserted.

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer</th>
<th>Application</th>
<th>XFS Command</th>
</tr>
</thead>
</table>
| 1.   | Select function cash-in. | Open the shutter of the input tray. | WFS_CMD_CIM_CASH_IN_START  
|      |          |             | WFS_CMD_CIM_OPEN_SHUTTER   |
| 2.   |          |             |             |
| 3.   |          |             | WFS_CMD_CIM_CLOSE_SHUTTER  
|      |          |             | WFS_CMD_CIM_CASH_IN (WFS_CIM_POSBILLINPUT) |
| 4.   | Insert money. |             | WFS_SRVE_CIM_ITEMSINSERTED  
|      |          |             | WFS_EXEE_CIM_INPUTP6 and completion of WFS_CMD_CIM_CASH_IN. |
| 5.   |          | Get number of level 2 / level 3 notes. | WFS_INF_CIM_GET_P6_INFO   |
| 6.   |          | Display the amount recognized so far and inform customer that level 2 / level 3 notes are inserted. |             |
| 7.   |          | Store signatures of level 2 / level 3 notes with customer data. | Call command WFS_INF_CIM_GET_P6_SIGNATURE once for every signature. |
| 8.   |          | Ask the customer for further actions:  
|      |          | If he wants to insert more money:   
|      |          | Repeat from step 2.   
|      |          | If he wants to finish the transaction:   
|      |          | Continue with step 9.   
|      |          | If he 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 customer's account, and inform the customer about the amounts of level 2 and level 3 notes. |             |
| 11.  |          | End of transaction. |             |
### 8.9 Multiple Refused Notes (Implicit Shutter Control)

The following table describes the flow of a cash-in transaction where items are rejected during the transaction and the Service Provider implicitly controls the shutter. In this case the `WFS_CMD_CIM_OPEN_SHUTTER` and `WFS_CMD_CIM_CLOSE_SHUTTER` commands are not used. Additionally, the number of items refused may be greater than the number of items that can be presented at the output position. Due to the complexity of this scenario, control of the shutter must be implicit. Therefore, there is no corresponding flow for explicit shutter control.

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer Action</th>
<th>Application</th>
<th>XFS Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.-5.</td>
<td>See OK Transaction (Implicit Shutter Control).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td></td>
<td>The Service Provider implicitly closes the shutter and begins bill recognition. As a result of the note processing n batches of notes must be returned to the customer.</td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td></td>
<td><code>WFS_EXEE_CIM_INPUTREFUSE</code></td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td></td>
<td>Return batch 1 of notes to customer. The Service Provider implicitly opens the shutter. <code>WFS_SRVE_CIM_ITEMSPRESENTED</code></td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td></td>
<td>Tell the customer that the bills were not accepted, and to take the bills.</td>
</tr>
<tr>
<td>10.</td>
<td>Customer removes unrecognized money.</td>
<td></td>
<td><code>WFS_SRVE_CIM_ITEMSTAKEN</code> The Service Provider implicitly closes the shutter.</td>
</tr>
<tr>
<td>11.</td>
<td></td>
<td></td>
<td>Repeat steps 11.-13. until batches 2 to n-1 are returned to the customer. The Service Provider implicitly opens the shutter. <code>WFS_SRVE_CIM_ITEMSPRESENTED</code></td>
</tr>
<tr>
<td>12.</td>
<td></td>
<td></td>
<td>Tell the customer to take the bills.</td>
</tr>
<tr>
<td>13.</td>
<td>Customer removes unrecognized money.</td>
<td></td>
<td><code>WFS_SRVE_CIM_ITEMSTAKEN</code> The Service Provider implicitly closes the shutter.</td>
</tr>
<tr>
<td>14.</td>
<td></td>
<td></td>
<td>Return Batch n (last) of notes to customer. The Service Provider implicitly opens the shutter. <code>WFS_SRVE_CIM_ITEMSPRESENTED</code></td>
</tr>
<tr>
<td>15.</td>
<td></td>
<td></td>
<td>Completion of <code>WFS_CMD_CIM_CASH_IN</code> with <code>WFS_SUCCESS</code>.</td>
</tr>
<tr>
<td>16.</td>
<td></td>
<td></td>
<td>Tell the customer to take the bills.</td>
</tr>
<tr>
<td>17.</td>
<td>Customer removes unrecognized money.</td>
<td></td>
<td><code>WFS_SRVE_CIM_ITEMSTAKEN</code> The Service Provider implicitly closes the shutter.</td>
</tr>
<tr>
<td>18.</td>
<td></td>
<td></td>
<td>Display the amount recognized so far.</td>
</tr>
<tr>
<td>19.</td>
<td></td>
<td></td>
<td>Ask the customer for further actions:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If he wants to deposit the amount: Continue with step 21.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>If he wants to get back all items inserted so far see table &quot;Cancellation by Customer (Implicit Shutter Control)&quot;</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>21.</td>
<td>Transport the money into the cash units RECYCLE_UNIT/CASHINBOX.</td>
<td>WFS_CMD_CIM_CASH_IN_END</td>
<td></td>
</tr>
<tr>
<td>22.</td>
<td>Credit the money to the customer’s account.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>23.</td>
<td>End of transaction.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.10 Multiple Rollback Notes (Implicit Shutter Control)

The following table describes the flow of a roll back operation where items are rolled back during the transaction and the Service Provider implicitly controls the shutter. In this case the WFS_CMD_CIM_OPEN_SHUTTER and WFS_CMD_CIM_CLOSE_SHUTTER commands are not used. Additionally, the number of items rolled back may be greater than the number of items that can be presented at the output position. Due to the complexity of this scenario, control of the shutter must be implicit. Therefore, there is no corresponding flow for explicit shutter control.

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer</th>
<th>Application</th>
<th>XFS Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.-9.</td>
<td>See Cancellation by Customer (Implicit Shutter Control).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td></td>
<td>Initiate the roll back operation.</td>
<td>WFS_CMD_CIM_CASH_IN_ROLLBACK</td>
</tr>
<tr>
<td>11.</td>
<td></td>
<td>The Service Provider begins the roll back. As a result of this n batches of notes must be returned to the customer.</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td></td>
<td>Return batch of notes to customer. The Service Provider implicitly opens the shutter. WFS_SRVE_CIM_ITEMSPRESENTED</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td></td>
<td>Tell the customer to take the bills.</td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td></td>
<td>Repeat steps 11.-14. until batches 2 to n-1 are returned to the customer.</td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td></td>
<td>Return batch n (last) of notes to customer. The Service Provider implicitly opens the shutter. WFS_SRVE_CIM_ITEMSPRESENTED</td>
<td></td>
</tr>
<tr>
<td>17.</td>
<td></td>
<td>Completion of WFS_CMD_CIM_CASH_IN_ROLLBACK with WFS_SUCCESS.</td>
<td></td>
</tr>
<tr>
<td>18.</td>
<td></td>
<td>Tell the customer to take the bills.</td>
<td></td>
</tr>
<tr>
<td>19.</td>
<td>Customer removes money.</td>
<td></td>
<td>WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter.</td>
</tr>
<tr>
<td>20.</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.11 Bill Recognition Error (WFS_CMD_CIM_PRESENT_MEDIA Command Supported)

The following table describes the flow of a cash-in transaction when the items are rejected as unrecognized during the transaction and the WFS_CMD_CIM_PRESENT_MEDIA command is supported.

This flow covers the following case:
- \( b\text{ShutterControl} = \text{FALSE}, b\text{PresentControl} = \text{FALSE}, b\text{ItemsTakenSensor} = \text{TRUE} \)

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer</th>
<th>Application</th>
<th>XFS Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.-6.</td>
<td>See OK Transaction (Explicit Shutter Control).</td>
<td></td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td></td>
<td>WFS_EXEE_CIM_INPUTREFUSE (InvalidBill) and completion of WFS_CMD_CIM_CASH_IN with WFS_SUCCESS.</td>
</tr>
<tr>
<td>8.</td>
<td>Present bills to customer.</td>
<td></td>
<td>WFS_CMD_CIM_PRESENT_MEDIA command is issued. The shutter opens, a WFS_SRVE_CIM_ITEMSPRESENTED is fired followed by a WFS_CMD_CIM_PRESENT_MEDIA command completion event.</td>
</tr>
<tr>
<td>9.</td>
<td>Tell the customer that the bills were not recognized and that he should take the bills.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>Customer removes unrecognized money.</td>
<td></td>
<td>WFS_SRVE_CIM_ITEMSTAKEN The Service Provider implicitly closes the shutter.</td>
</tr>
<tr>
<td>11.</td>
<td>Display the amount recognized so far.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>Ask the customer for further actions:</td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN_END</td>
</tr>
<tr>
<td>13.</td>
<td>If he wants to deposit the amount: Continue with step 14.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>Transport the money into the cash units RECYCLE_UNIT/CASHINBOX.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>Credit the money to the customer's account.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td>End of transaction.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
8.12 Cancellation by Customer (Implicit Shutter Control and WFS_CMD_CIM_PRESENT_MEDIA Command Supported)

The following table describes the flow of a cash-in transaction where the customer wants all the items to be returned after recognition and the WFS_CMD_CIM_PRESENT_MEDIA command is supported.

This flow covers the following case:

- \( b_{\text{ShutterControl}} = \text{TRUE}, b_{\text{PresentControl}} = \text{FALSE}, b_{\text{ItemsTakenSensor}} = \text{TRUE} \)

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer</th>
<th>Application</th>
<th>XFS Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-9.</td>
<td>See OK Transaction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td>Selection : Return all the items.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td>Transport the items recognized to an internal position.</td>
<td>WFS_CMD_CIM_CASH_IN_ROLLBACK</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN_ROLLBACK completion.</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>Present bills to the customer.</td>
<td>WFS_CMD_CIM_PRESENT_MEDIA command is issued. The shutter opens and a WFS_SRVE_CIM_ITEMSPRESENTED is fired.</td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td></td>
<td>WFS_CMD_CIM_PRESENT_MEDIA completion event.</td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>Request removal of the money.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td>Customer takes the money from the output position.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>17.</td>
<td></td>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td></td>
</tr>
<tr>
<td>18.</td>
<td></td>
<td>End of transaction.</td>
<td></td>
</tr>
</tbody>
</table>
Compound CIM/IPM deposit devices are able to accept and process different types of media such as cash and checks. In order to improve the speed and usability of deposit devices it may be desirable to allow a bunch of items deposited to contain a variety of media types. Typically this is a bunch containing both cash and checks and is termed ‘Mixed Media processing’.

During this type of transaction the customer will insert cash and checks together in one bunch. The device will identify each item. Items not positively identified may be immediately returned to the customer. All remaining items can be deposited and shared deposit bins can be configured to receive mixed items. The application can also choose to return all items. Additionally the specification allows for depositing all checks and returning all cash or vice-versa depending on requirements.

In order to facilitate devices of differing hardware design and to support reuse of the XFS API, Mixed Media processing is achieved by initiating a CIM and an IPM transaction in parallel. The application and Service Providers must be able to handle concurrent CIM and IPM commands and events. The application will use the WFS_CMD_CIM_SET_MODE or WFS_CMD_IPM_SET_MODE command to activate Mixed Media processing. The literals used (i.e. WFS_CIM_IPMMIXEDMEDIA) describe the modes and indicate the nature of the compound device. This allows applications to open the correct interfaces to drive the transaction.

Mixed Media processing commands that move media in the device require commands to be called on both CIM and IPM interfaces. See the table below for a list of CIM commands and their IPM counterparts. Where the operation is to be cancelled the application is required to cancel only one command on either the CIM or IPM interface. Applications must be aware that the command that was NOT explicitly cancelled may complete with a WFS_ERR_CANCELED error.

For example the application must call both WFS_CMD_CIM_CASH_IN and WFS_CMD_IPM_MEDIA_IN commands to initiate the transaction. If an application wishes to cancel the transaction before items are inserted, only the WFS_CMD_CIM_CASH_IN command can be cancelled and the WFS_CMD_IPM_MEDIA_IN command will also be cancelled.

Devices suitable for Mixed Media processing must report WFSCIMCAPS.\text{bShutterControl} == TRUE to allow WFS_CMD_CIM_PRESENT_MEDIA and WFS_CMD_IPM_PRESENT_MEDIA commands to work concurrently.

The Mixed Media mode can be determined by calling WFS_INF_CIM_STATUS or WFS_INF_IPM_STATUS command and checking the value of the \text{wMixedMode} field.

Where an error occurs both CIM and IPM interfaces will report it. To recover the device a reset command can be called on either of the interfaces. Reset calls on both CIM and IPM interfaces are not required.

Application refusal (in the IPM interface) is not supported in Mixed Media mode.

To initiate a Mixed Media transaction the WFS_CMD_CIM_CASH_IN_START command must be called. There is no equivalent command to the WFS_CMD_CIM_CASH_IN_START command on the IPM interface.

<table>
<thead>
<tr>
<th>CIM command</th>
<th>IPM Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CMD_CIM_CASH_IN</td>
<td>WFS_CMD_IPM_MEDIA_IN</td>
</tr>
<tr>
<td>WFS_CMD_CIM_CASH_IN_END</td>
<td>WFS_CMD_IPM_MEDIA_IN_END or where bMixedDepositAndRollback is TRUE</td>
</tr>
<tr>
<td>WFS_CMD_CIM_CASH_IN_ROLLBACK</td>
<td>WFS_CMD_IPM_MEDIA_IN_ROLLBACK or where bMixedDepositAndRollback is TRUE</td>
</tr>
<tr>
<td>WFS_CMD_CIM_PRESENT_MEDIA</td>
<td>WFS_CMD_IPM_PRESENT_MEDIA</td>
</tr>
<tr>
<td>WFS_CMD_CIM_RETRACT</td>
<td>WFS_CMD_IPM_RETRACT_MEDIA</td>
</tr>
</tbody>
</table>

Application refusal (in the IPM interface) is not supported in Mixed Media mode.

To initiate a Mixed Media transaction the WFS_CMD_CIM_CASH_IN_START command must be called. There is no equivalent command to the WFS_CMD_CIM_CASH_IN_START command on the IPM interface.

**Commands and their counterparts:**

This table lists the counterpart IPM commands which must be called as well as the CIM commands when in Mixed Media processing mode.
Events and their Counterparts
The CIM and IPM interfaces both have a range of events to inform the application of device activity. During Mixed Media processing events fired from each interface can describe the same situation (i.e. items presented). In these cases the recommendation to application developers is to rely on a single interface for these duplicate notifications. The choice of which interface to use to handle specific events will be based on factors such as current codebase or application presentation requirements.

<table>
<thead>
<tr>
<th>CIM Event</th>
<th>IPM Event</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_USRE_CIM_CASHUNITTHRESHOLD</td>
<td>WFS_USRE_IPM_MEDIABINTHRESHOLD</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_CASHUNITINFOCHANGED</td>
<td>WFS_SRVE_IPM_MEDIABININFOCHANGED</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_CASHUNITERROR</td>
<td>WFS_EXEE_IPM_MEDIABINERROR</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>WFS_SRVE_IPM_MEDIATAKEN</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_COUNTS_CHANGED</td>
<td>WFS_SRVE_IPM_MEDIABININFOCHANGED</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INPUTREFUSE</td>
<td>WFS_EXEE_IPM_MEDIAREFUSED</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSPRESENTED</td>
<td>WFS_EXEE_IPM_MEDIAPRESENTED</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_ITEMSINSERTED</td>
<td>WFS_EXEE_IPM_MEDIAINSERTED</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_SUBCASHIN</td>
<td>WFS_EXEE_IPM_MEDIADATA</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_MEDIADETECTED</td>
<td>WFS_SRVE_IPM_MEDIADETECTED</td>
</tr>
<tr>
<td>WFS_EXEE_CIM_INSERTITEMS</td>
<td>WFS_EXEE_IPM_NOMEDIA</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_DEVICEPOSITION</td>
<td>WFS_SRVE_IPM_DEVICEPOSITION</td>
</tr>
<tr>
<td>WFS_SRVE_CIM_POWER_SAVE_CHANGE</td>
<td>WFS_SRVE_IPM_POWER_SAVE_CHANGE</td>
</tr>
</tbody>
</table>

The following sections describe the flow of a Mixed Media transaction on a compound CIM/IPM device. These application flows are provided as guidelines only. In all cases WFSCIMPOS.CAPS.bPresentControl == TRUE unless otherwise stated.
### 9.1 Mixed Media OK Transaction

The following table describes a normal Mixed Media transaction flow where there is a successful deposit.

This flow covers the following case:

- $b\text{ShutterControl} = \text{TRUE}$, $w\text{MixedMode} = \text{WFS\_CIM\_IPMMIXEDMEDIA}$

<table>
<thead>
<tr>
<th>Step</th>
<th>Application/Customer</th>
<th>CIM Commands and Events</th>
<th>IPM Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Application transaction opens sessions with both the CIM and the IPM service providers.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>Customer selects Mixed Media transaction.</td>
<td>WFS_CMD_CIM_CASH_IN_START</td>
<td>WFS_CMD_IPM_MEDIA_IN (Service Provider opens the input shutter).</td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td>WFS_CMD_IPM_MEDIA_IN (The shutter is not opened until WFS_CMD_IPM_MEDIA_IN called.)</td>
<td>WFS_EXEE_IPM_NOMEDIA This event specifies that media must be inserted into the device in order for the execute command to proceed.</td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td>WFS_EXEE_CIM_INSERTITEMS event is sent when the shutter is fully open and the device is ready to begin accepting items.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td>Ask the customer to insert items.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td>Customer inserts items.</td>
<td>WFS_SRVE_CIM_ITEMSINSERTED</td>
<td>WFS_EXEE_IPM_MEDIA_INSERTED</td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td>The Service Provider closes the input shutter and the device begins processing the inserted items.</td>
<td>Send one WFS_EXEE_IPM_MEDIADATA event for every check item identified.</td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td>The WFS_CMD_CIM_CASH_IN command completes.</td>
<td>The WFS_CMD_IPM_MEDIA_IN command completes.</td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td>WFS_INF_CIM_CASH_IN_STATUS can be issued to request the number of CIM related items that were inserted.</td>
<td>WFS_INF_IPM_TRANSACTION_STATUS is issued to request the number of IPM related items that were inserted.</td>
</tr>
<tr>
<td>10.</td>
<td>Display the items recognized and associated information so far.</td>
<td></td>
<td>Process the checks by sending any of: WFS_CMD_IPM_READ_IMAGE, WFS_CMD_IPM_SET_DESTINATION, WFS_CMD_IPM_PRINT_TEXT, WFS_CMD_IPM_GET_IMAGE_AFTER_PRINT</td>
</tr>
<tr>
<td>11.</td>
<td>Ask the customer for further actions:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td>If he wants to insert more items: Repeat from step 3.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If he wants to finish the transaction: Continue with step 13.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>If he wants to get back all items inserted so far see table &quot;Cancellation by Customer&quot;.</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>WFS_CMD_CIM_CASH_IN_END (The device will not complete the media movement until WFS_CMD_IPM_MEDIA_IN_END command is called on IPM interface.)</td>
<td>WFS_CMD_IPM_MEDIA_IN_END Print on individual media items (as specified from IPM commands).</td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>Transport the items into the specified destinations.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>WFS_CMD_CIM_CASH_IN_END completes.</td>
<td>WFS_CMD_IPM_MEDIA_IN_END completes. Output parameter indicates media bin / outputs positions that have received items.</td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td>Credit the appropriate funds to the customer's account.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td>End of transaction.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 9.2 Mixed Media Cancellation by Customer

The following table describes the flow of a Mixed Media transaction where the customer wants all the items to be returned. In this case the returned items must be explicitly presented by the application.

This flow covers the following cases:

- $b_{\text{itemsInsertedSensor}} = \text{TRUE}$, $b_{\text{itemsTakenSensor}} = \text{TRUE}$
- $b_{\text{Compound}} = \text{TRUE}$, $w_{\text{MixedMode}} = \text{WFS_CIM_IPMMIXEDMEDIA}$
- $WFSCIMPOSCAPS.b_{\text{PresentControl}} = \text{FALSE}$

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer/Application</th>
<th>CIM Commands and Events</th>
<th>IPM Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.- 12.</td>
<td>As per OK Transaction.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td>Selection: Return all the items.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>Transport the items recognized to the output position.</td>
<td>WFS_CMD_CIM_CASH_IN_-_ROLLBACK (No physical action may take place until the WFS_CMD_IPM_MEDIA_IN_ROLLBACK command.)</td>
<td>WFS_CMD_IPM_MEDIA_IN_-_ROLLBACK</td>
</tr>
<tr>
<td>15.</td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN_-_ROLLBACK completion.</td>
<td>WFS_CMD_IPM_MEDIA_IN_-_ROLLBACK completion.</td>
</tr>
<tr>
<td>16.</td>
<td></td>
<td>WFS_CMD_CIM_PRESENT_MEDIA (No physical action may take place until the WFS_CMD_IPM_PRESENT_MEDIA command.)</td>
<td>WFS_CMD_IPM_PRESENT_MEDIA</td>
</tr>
<tr>
<td>17.</td>
<td>The Service Provider opens the shutter(s). CIM cash moves to output position.</td>
<td>IPM media moves to output position.</td>
<td></td>
</tr>
<tr>
<td>19.</td>
<td></td>
<td>WFS_CMD_CIM_PRESENT_MEDIA completes.</td>
<td>WFS_CMD_IPM_PRESENT_MEDIA completes.</td>
</tr>
<tr>
<td>20.</td>
<td>Customer takes the items from the output position.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>21.</td>
<td></td>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>WFS_SRVE_IPM_MEDIATAKEN</td>
</tr>
<tr>
<td>22.</td>
<td></td>
<td>The Service Provider closes the shutter.</td>
<td></td>
</tr>
<tr>
<td>23.</td>
<td>End of transaction.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
### 9.3 Mixed Media Cancellation by Customer on Cash Part Only

The following table describes the flow of a Mixed Media transaction where the customer wants the cash items to be returned but deposit the check items. In this case the returned items are implicitly presented by the Service Provider.

This flow covers the following cases:

- \( b\text{ItemsInsertedSensor} == \text{TRUE} \), \( b\text{ItemsTakenSensor} == \text{TRUE} \)
- \( w\text{MixedMode} == \text{WFS_CIM_IPMMIXEDMEDIA} \)
- \( \text{WFSCIMPOSCAPS.bPresentControl} == \text{TRUE} \)

<table>
<thead>
<tr>
<th>Step</th>
<th>Customer/ Application</th>
<th>CIM Commands and Events</th>
<th>IPM Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.-12.</td>
<td>As per OK transaction</td>
<td></td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td>Transport the items recognized to the output position.</td>
<td>WFS_CMD_CIM_CASH_IN_-ROLLBACK (No physical action may take place until the WFS_CMD_IPM_MEDIA_IN_END command.)</td>
<td>WFS_CMD_IPM_MEDIA_IN_END</td>
</tr>
<tr>
<td>15.</td>
<td></td>
<td>Print on, and deposit individual media items (as specified by IPM commands).</td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td></td>
<td>The Service Provider opens the shutter. CIM cash moves to output position.</td>
<td></td>
</tr>
<tr>
<td>18.</td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN_-ROLLBACK completes.</td>
<td>WFS_CMD_IPM_MEDIA_IN_END completes.</td>
</tr>
<tr>
<td>19.</td>
<td>Customer takes the cash items from the output position.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>20.</td>
<td></td>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>WFS_SRVE_IPM_MEDIATAKEN</td>
</tr>
</tbody>
</table>
9.4 Mixed Media Multiple Refused Items

The following table describes the flow of a Mixed Media transaction where items are rejected during the transaction. Additionally, the number of items refused may be greater than the number of items that can be presented at the output position. In this case the returned items must be explicitly presented by the application.

This flow covers the following cases:

- \( b\text{ShutterControl} == \text{TRUE} \), \( b\text{ItemsInsertedSensor} == \text{TRUE} \), \( b\text{ItemsTakenSensor} == \text{TRUE} \)
- \( b\text{Compound} == \text{TRUE} \), \( w\text{MixedMode} == \text{WFS\_CIM\_IPMMIXEDMEDIA} \)
- \( \text{WFSCIMPOSCAPS}.b\text{PresentControl} == \text{FALSE} \)

<table>
<thead>
<tr>
<th>Step</th>
<th>Application/ Customer</th>
<th>CIM Commands and Events</th>
<th>IPM Commands and Events</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Customer selects Mixed Media transaction.</td>
<td>WFS_CMD_CIM_CASH_IN_START</td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td></td>
<td>WFS_CMD_CIM_CASH_IN (The shutter is not opened until WFS_CMD_IPM_MEDIA_IN called.)</td>
<td>WFS_CMD_IPM_MEDIA_IN</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Service Provider opens the input shutter.</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td></td>
<td>WFS_EXEE_CIM_INSERTITEMS event is sent when the shutter is fully open and the device is ready to begin accepting items.</td>
<td>WFS_EXEE_IPM_NOMEDIA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>This event specifies that media must be inserted into the device in order for the execute command to proceed.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td></td>
<td>Ask the customer to insert items.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td></td>
<td>Customer inserts items.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td></td>
<td>WFS_SRVE_CIM_ITEMSINSERTED</td>
<td>WFS_EXEE_IPM_MEDIAINserted</td>
</tr>
<tr>
<td>7.</td>
<td></td>
<td>The Service Provider closes the input shutter and the device begins processing the inserted items.</td>
<td>Send one WFS_EXEE_IPM_MEDIADATA event for every check item identified.</td>
</tr>
<tr>
<td>8.</td>
<td></td>
<td>Items are refused. WFS_EXEE_CIM_INPUTREFUSE event sent with appropriate ( lp\text{usReason} ) parameter. Items that are not bills or checks are rejected with WFS_CIM_INVALIDBILL.</td>
<td>WFS_EXEE_IPM_MEDIAREFUSED</td>
</tr>
<tr>
<td>9.</td>
<td></td>
<td>The WFS_CMD_CIM_CASH_IN command completes.</td>
<td>WFS_CMD_IPM_MEDIA_IN command completes.</td>
</tr>
<tr>
<td>10.</td>
<td>Application chooses to return refused items now.</td>
<td>WFS_CMD_CIM_PRESENT_MEDIA (No physical action may take place until the WFS_CMD_IPM_PRESENT_MEDIA command.)</td>
<td>WFS_CMD_IPM_PRESENT_MEDIA</td>
</tr>
<tr>
<td>11.</td>
<td>Each bunch of items presented.</td>
<td>WFS_SRVE_CIM_ITEMSPRESENTED</td>
<td>WFS_EXEE_IPM_MEDIAPRESENTED</td>
</tr>
<tr>
<td>12.</td>
<td>All but last bunch of items taken.</td>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>WFS_SRVE_IPM_MEDIATAKEN</td>
</tr>
<tr>
<td>13.</td>
<td></td>
<td>WFS_CMD_CIM_PRESENT_MEDIA command completes.</td>
<td>WFS_CMD_IPM_PRESENT_MEDIA command completes.</td>
</tr>
<tr>
<td>14.</td>
<td>Last bunch of items taken.</td>
<td>WFS_SRVE_CIM_ITEMSTAKEN</td>
<td>WFS_SRVE_IPM_MEDIATAKEN</td>
</tr>
<tr>
<td>15.</td>
<td>Transaction continues from step 13. in the OK transaction.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>


10. Rules for Cash Unit Exchange

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

If a cash unit is removed from the CIM outside of the Start/End Exchange operations and subsequently reinserted the status of the physical cash unit should be set to WFS_CIM_STATCUMANIP to indicate to the application that the physical cash unit has been removed, reinserted and possibly tampered with. While the cash unit has this status the Service Provider should not attempt to use it as part of a cash-in operation. The WFS_CIM_STATCUMANIP status should not change until the next Start/End Exchange operation is performed, even if the cash unit is replaced in its original position.

If all the physical cash units belonging to a logical cash unit are manipulated the parent logical cash unit that the physical cash units belong to should also have its status set to WFS_CIM_STATCUMANIP.

When a cash unit is removed and/or replaced outside of the Start/End Exchange operations the original logical cash unit information such as the values, currency and counts should be preserved in the Cash Unit Info structure reported to the application for accounting purposes until the next Start/End Exchange operations, even if the cash unit physically contains a different denomination.

**Mixed Media Processing:**

Where the device supports cash units that can store non-CIM items, a counters update to those cash units applied by the CIM interface can also be seen in the other interfaces available to the compound device.

The CIM ulCount on a shared bin (of type WFS_CIM_TYPECASHIN) reports the total number of banknotes, checks or coins of all types in the cash unit. This is for the following reasons:

1. ulCount on CIM has the same meaning as ulCount on IPM. That is the number of items of any type in the bin.
2. ulMaximum, is truly representative of the capacity of the physical bin and software thresholds can accurately reflect the state of the bin.
3. Use of ulCount representing items from both interfaces gives the greatest flexibility. Dedicated CIM or IPM bins and therefore counts can still be achieved through bin configuration.
4. The actual number of notes can be determined from lpNoteNumberList.

The following table describes the effect on the IPM counts where an application causes counter changes to a shared cassette using the CIM interface. The example assumes the starting position of a shared CIM cash unit/IPM media bin:

<table>
<thead>
<tr>
<th>Application Activity</th>
<th>CIM Counts on the shared cash unit</th>
<th>IPM Counts on the shared media bin</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. A customer enters 10 good notes and 10 good checks in the same transaction.</td>
<td>ulCashInCount = 10 ulCount = 20</td>
<td>ulMediaInCount = 10 ulCount = 20</td>
</tr>
<tr>
<td>2. Replenishment activity removes all items from the cash unit and clears the counts using WFS_CMD_CIM_END_EXCHANGE</td>
<td>ulCashInCount = 0 ulCount = 0</td>
<td>ulMediaInCount = 0 ulCount = 0</td>
</tr>
<tr>
<td></td>
<td>Description</td>
<td>ulCashInCount</td>
</tr>
<tr>
<td>---</td>
<td>-------------</td>
<td>---------------</td>
</tr>
<tr>
<td>3.</td>
<td>A further customer enters 10 good notes and 10 good checks in the same transaction.</td>
<td>10</td>
</tr>
<tr>
<td>4.</td>
<td>Replenishment activity removes only cash items from the cash unit. The CIM counts are adjusted using WFS_CMD_CIM_SET_CASH_UNIT_INFO. ulCashInCount is set to 0, and ulCount is set to 10.</td>
<td>0</td>
</tr>
<tr>
<td>5.</td>
<td>A further customer enters 10 good notes and 10 good checks in the same transaction.</td>
<td>10</td>
</tr>
<tr>
<td>6.</td>
<td>Replenishment activity removes only checks (20 items) from the cash unit. The counts are adjusted using WFS_CMD_IPM_SET_MEDIA_BIN_INFO. ulMediaInCount is set to 0, and ulCount is set to 10.</td>
<td>10</td>
</tr>
</tbody>
</table>
#ifndef __INC_XFSCIM__H
#define __INC_XFSCIM__H

#endif __cplusplus
#endif __cplusplus

#include <xfsapi.h>

#pragma pack (push, 1)

/* values of WFSCIMCAPS.wClass */

#define WFS_SERVICE_CLASS_CIM (13)
#define WFS_SERVICE_CLASS_VERSION_CIM (0x1403) /* Version 3.20 */
#define WFS_SERVICE_CLASS_NAME_CIM "CIM"

#define CIM_SERVICE_OFFSET (WFS_SERVICE_CLASS_CIM * 100)

/* CIM Info Commands */

#define WFS_INF_CIM_STATUS (CIM_SERVICE_OFFSET + 1)
#define WFS_INF_CIM_CAPABILITIES (CIM_SERVICE_OFFSET + 2)
#define WFS_INF_CIM_CASH_UNIT_INFO (CIM_SERVICE_OFFSET + 3)
#define WFS_INF_CIM_TELLER_INFO (CIM_SERVICE_OFFSET + 4)
#define WFS_INF_CIM_CURRENCY_EXP (CIM_SERVICE_OFFSET + 5)
#define WFS_INF_CIM_BANKNOTE_TYPES (CIM_SERVICE_OFFSET + 6)
#define WFS_INF_CIM_CASH_IN_STATUS (CIM_SERVICE_OFFSET + 7)
#define WFS_INF_CIM_GET_P6_INFO (CIM_SERVICE_OFFSET + 8)
#define WFS_INF_CIM_GET_P6_SIGNATURE (CIM_SERVICE_OFFSET + 9)
#define WFS_INF_CIM_GET_ITEM_INFO (CIM_SERVICE_OFFSET + 10)
#define WFS_INF_CIM_POSITION_CAPABILITIES (CIM_SERVICE_OFFSET + 11)
#define WFS_INF_CIM_REPLENISH_TARGET (CIM_SERVICE_OFFSET + 12)
#define WFS_INF_CIM_DEVICELOCK_STATUS (CIM_SERVICE_OFFSET + 13)
#define WFS_INF_CIM_CASH_UNIT_CAPABILITIES (CIM_SERVICE_OFFSET + 14)

/* CIM Execute Commands */

#define WFS_CMD_CIM_CASH_IN_START (CIM_SERVICE_OFFSET + 1)
#define WFS_CMD_CIM_CASH_IN (CIM_SERVICE_OFFSET + 2)
#define WFS_CMD_CIM_CASH_IN_END (CIM_SERVICE_OFFSET + 3)
#define WFS_CMD_CIM_CASH_IN_ROLLBACK (CIM_SERVICE_OFFSET + 4)
#define WFS_CMD_CIM_RETRACT (CIM_SERVICE_OFFSET + 5)
#define WFS_CMD_CIM_OPEN_SHUTTER (CIM_SERVICE_OFFSET + 6)
#define WFS_CMD_CIM_CLOSE_SHUTTER (CIM_SERVICE_OFFSET + 7)
#define WFS_CMD_CIM_SET_TELLER_INFO (CIM_SERVICE_OFFSET + 8)
#define WFS_CMD_CIM_SET_CASH_UNIT_INFO (CIM_SERVICE_OFFSET + 9)
#define WFS_CMD_CIM_START_EXCHANGE (CIM_SERVICE_OFFSET + 10)
#define WFS_CMD_CIM_END_EXCHANGE (CIM_SERVICE_OFFSET + 11)
#define WFS_CMD_CIM_OPEN_SAFE_DOOR (CIM_SERVICE_OFFSET + 12)
#define WFS_CMD_CIM_RESET (CIM_SERVICE_OFFSET + 13)
#define WFS_CMD_CIM_CONFIGURE_CASH_IN_UNITS (CIM_SERVICE_OFFSET + 14)
#define WFS_CMD_CIM_CONFIGURE_NOTETYPES (CIM_SERVICE_OFFSET + 15)
#define WFS_CMD_CIM_CREATE_P6_SIGNATURE (CIM_SERVICE_OFFSET + 16)
#define WFS_CMD_CIM_SET_GUIDANCE_LIGHT (CIM_SERVICE_OFFSET + 17)
#define WFS_CMD_CIM_CONFIGURE_NOTE_READER (CIM_SERVICE_OFFSET + 18)
#define WFS_CMD_CIM_COMPARE_P6_SIGNATURE (CIM_SERVICE_OFFSET + 19)
#define WFS_CMD_CIM_POWER_SAVE_CONTROL (CIM_SERVICE_OFFSET + 20)
#define WFS_CMD_CIM_REPLENISH (CIM_SERVICE_OFFSET + 21)
#define WFS_CMD_CIM_SET_CASH_IN_LIMIT (CIM_SERVICE_OFFSET + 22)
#define WFS_CMD_CIM_CASH_UNIT_COUNT (CIM_SERVICE_OFFSET + 23)
#define WFS_CMD_CIM_DEVICE_LOCK_CONTROL (CIM_SERVICE_OFFSET + 24)
#define WFS_CMD_CIM_SET_MODE (CIM_SERVICE_OFFSET + 25)
#define WFS_CMD_CIM_PRESENT_MEDIA (CIM_SERVICE_OFFSET + 26)

/* CIM Messages */

#define WFS_SRV_CE_CIM_SAFEDOOROPEN (CIM_SERVICE_OFFSET + 1)
#define WFS_SRV_CE_CIM_SAFEDOORCLOSED (CIM_SERVICE_OFFSET + 2)
#define WFS_USR_CE_CIM_CASHUNITTHRESHOLD (CIM_SERVICE_OFFSET + 3)
#define WFS_SRV_CE_CIM_CASHUNITINFOCHANGED (CIM_SERVICE_OFFSET + 4)
#define WFS_SRV_CE_CIM_TELLERINFOCHANGED (CIM_SERVICE_OFFSET + 5)
#define WFS_EXE_CE_CIM_CASHUNITERROR (CIM_SERVICE_OFFSET + 6)
#define WFS_SRV_CE_CIM_ITEMSTAKEN (CIM_SERVICE_OFFSET + 7)
#define WFS_SRV_CE_CIM_COUNTS_CHANGED (CIM_SERVICE_OFFSET + 8)
#define WFS_EXE_CE_CIM_INPUTREFUSE (CIM_SERVICE_OFFSET + 9)
#define WFS_SRV_CE_CIM_ITEMSPRESENTED (CIM_SERVICE_OFFSET + 10)
#define WFS_SRV_CE_CIM_ITEMSINSERTED (CIM_SERVICE_OFFSET + 11)
#define WFS_EXE_CE_CIM_NOTERROR (CIM_SERVICE_OFFSET + 12)
#define WFS_SRV_CE_CIM_MEDIADETECTED (CIM_SERVICE_OFFSET + 14)
#define WFS_EXE_CE_CIM_INPUT_P6 (CIM_SERVICE_OFFSET + 15)
#define WFS_EXE_CE_CIM_INFO_AVAILABLE (CIM_SERVICE_OFFSET + 16)
#define WFS_EXE_CE_CIM_INSERTITEMS (CIM_SERVICE_OFFSET + 17)
#define WFS_SRV_CE_CIM_DEVICEPOSITION (CIM_SERVICE_OFFSET + 18)
#define WFS_SRV_CE_CIM_POWER_SAVE_CHANGE (CIM_SERVICE_OFFSET + 19)
#define WFS_EXE_CE_CIM_INCOMPLETEREPLENISH (CIM_SERVICE_OFFSET + 20)

/* values of WFSCIMSTATUS.fwDevice */

#define WFS_CIM_DEVONLINE WFS_STAT_DEVONLINE
#define WFS_CIM_DEVOFFLINE WFS_STAT_DEVOFFLINE
#define WFS_CIM_DEVPOWEROFF WFS_STAT_DEVPOWEROFF
#define WFS_CIM_DEVNODEVICE WFS_STAT_DEVNODEVICE
#define WFS_CIM_DEVUSERERROR WFS_STAT_DEVUSERERROR
#define WFS_CIM_DEVHWERROR WFS_STAT_DEVHWERROR
#define WFS_CIM_DEVBUSY WFS_STAT_DEVBUSY
#define WFS_CIM_DEVFRAUDATTEMPT WFS_STAT_DEVFRAUDATTEMPT
#define WFS_CIM_DEVPOWEROFF WFS_STAT_DEVPOWEROFF
#define WFS_CIM_DEVFRAUDATTEMPT WFS_STAT_DEVFRAUDATTEMPT

/* values of WFSCIMSTATUS.fwSafeDoor */

#define WFS_CIM_DOORNOTSUPPORTED (1)
#define WFS_CIM_DOOROPEN (2)
#define WFS_CIM_DOORCLOSED (3)
#define WFS_CIM_DOORUNKNOWN (4)

/* values of WFSCIMSTATUS.fwAcceptor */

#define WFS_CIM_ACCOK (0)
#define WFS_CIM_ACCUSTATE (1)
#define WFS_CIM_ACCUSTOP (2)
#define WFS_CIM_ACCUNKNOWN (3)

/* values of WFSCIMSTATUS.fwIntermediateStacker */

#define WFS_CIM_ISEMPTY (0)
#define WFS_CIM_ISNOTEMPTY (1)
#define WFS_CIM_ISFULL (2)
#define WFS_CIM_ISUNKNOWN (4)
#define WFS_CIM_ISNOTSUPPORTED (5)

/* Size and max index of dwGuidLights array */

#define WFS_CIM_GUIDLIGHTS_SIZE (32)
#define WFS_CIM_GUIDLIGHTS_MAX (WFS_CIM_GUIDLIGHTS_SIZE - 1)

/* Indices of WFSCIMSTATUS.dwGuidLights [...] */

WFS_CIMCAPS.dwGuidLights [...]

143
#define WFS_CIM_GUIDANCE_POSINNULL 0
#define WFS_CIM_GUIDANCE_POSINLEFT 1
#define WFS_CIM_GUIDANCE_POSINRIGHT 2
#define WFS_CIM_GUIDANCE_POSINCENTER 3
#define WFS_CIM_GUIDANCE_POSINTOP 4
#define WFS_CIM_GUIDANCE_POSINBOTTOM 5
#define WFS_CIM_GUIDANCE_POSINFRONT 6
#define WFS_CIM_GUIDANCE_POSINREAR 7
#define WFS_CIM_GUIDANCE_POSOUTLEFT 8
#define WFS_CIM_GUIDANCE_POSOUTRIGHT 9
#define WFS_CIM_GUIDANCE_POSOUTCENTER 10
#define WFS_CIM_GUIDANCE_POSOUTTOP 11
#define WFS_CIM_GUIDANCE_POSOUTBOTTOM 12
#define WFS_CIM_GUIDANCE_POSOUTFRONT 13
#define WFS_CIM_GUIDANCE_POSOUTREAR 14
#define WFS_CIM_GUIDANCE_POSOUTNULL 15

/* Values of WFSCIMSTATUS.dwGuidLights */
WFCSCIMSTATUS.dwGuidLights [ ]

#define WFS_CIM_GUIDANCE_NOT_AVAILABLE 0x00000000
#define WFS_CIM_GUIDANCE_OFF 0x00000001
#define WFS_CIM_GUIDANCE_SLOW_FLASH 0x00000004
#define WFS_CIM_GUIDANCE_MEDIUM_FLASH 0x00000008
#define WFS_CIM_GUIDANCE_QUICK_FLASH 0x00000010
#define WFS_CIM_GUIDANCE_CONTINUOUS 0x00000080
#define WFS_CIM_GUIDANCE_RED 0x00000100
#define WFS_CIM_GUIDANCE_GREEN 0x00000200
#define WFS_CIM_GUIDANCE_YELLOW 0x00000400
#define WFS_CIM_GUIDANCE_BLUE 0x00000800
#define WFS_CIM_GUIDANCE_CYAN 0x00001000
#define WFS_CIM_GUIDANCE_MAGENTA 0x00002000
#define WFS_CIM_GUIDANCE_WHITE 0x00004000

/* values of WFSCIMSTATUS.wDevicePosition */
WFCSCIMDEVICEPOSITION.wPosition

#define WFS_CIM_DEVICEINPOSITION 0
#define WFS_CIM_DEVICENOTINPOSITION 1
#define WFS_CIM_DEVICEPOSUNKNOWN 2
#define WFS_CIM_DEVICEPOSNOTSUPP 3

/* values of WFSCIMSTATUS.fwStackerItems */

#define WFS_CIM_CUSTOMERACCESS 0
#define WFS_CIM_NOCUSTOMERACCESS 1
#define WFS_CIM_ACCESSUNKNOWN 2
#define WFS_CIM_NOITEMS 4

/* values of WFSCIMSTATUS.fwBankNoteReader */

#define WFS_CIM_BNROK 0
#define WFS_CIM_BNRINOP 1
#define WFS_CIM_BNRUNKNOWN 2
#define WFS_CIM_BNRNOTSUPPORTED 3

/* values of WFSCIMSTATUS.fwShutter */

#define WFS_CIM_SHTCLOSED 0
#define WFS_CIM_SHTOPEN 1
#define WFS_CIM_SHTJAMMED 2
#define WFS_CIM_SHTUNKNOWN 3
#define WFS_CIM_SHTNOTSUPPORTED 4

/* values of WFSCIMCAPS.wMixedMode */

#define WFS_CIM_MIXEDMEDIANOTSUPP 0
#define WFS_CIM_IPMMIXEDMEDIA (1)
/* values of WFSCIMSETMODE.wMixedMode */
/* values of WFSCIMSTATUS.wMixedMode.*/

#define WFS_CIM_MIXEDMEDIANOTACTIVE (0)
/* values of WFSCIMINPOS.fwPositionStatus */
#define WFS_CIM_PSEMPTY (0)
#define WFS_CIM_PSNOTEMPTY (1)
#define WFS_CIM_PSUNKNOWN (2)
#define WFS_CIM_PSNOTSUPPORTED (3)
#define WFS_CIM_PSFOREIGNITEMS (4)
/* values of WFSCIMSTATUS.fwTransport */
#define WFS_CIM_TPOK (0)
#define WFS_CIM_TPINOP (1)
#define WFS_CIM_TPUNKNOWN (2)
#define WFS_CIM_TPNOTSUPPORTED (3)
/* values of WFSCIMINPOS.fwTransportStatus */
#define WFS_CIM_TPSTATEMPTY (0)
#define WFS_CIM_TPSTATNOTEMPTY (1)
#define WFS_CIM_TPSTATNOTEMPTYCUST (2)
#define WFS_CIM_TPSTATNOTEMPTY_UNK (3)
#define WFS_CIM_TPSTATNOTSUPPORTED (4)
/* values of 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_RETRACTCASSETTE (0x0004)
#define WFS_CIM_REJECT (0x0008)
#define WFS_CIM_BILLCASSETTES (0x0010)
/* values for WFSCIMCAPS.fwCashInLimit */
#define WFS_CIM_LIMITNOTSUPP (0x0000)
#define WFS_CIM_LIMITBYTOTALITEMS (0x0001)
#define WFS_CIM_LIMITBYAMOUNT (0x0002)
/* values of WFSCIMCASHIN.fwType */
#define WFS_CIM_TYPERECYCLING (1)
#define WFS_CIM_TYPECASHIN (2)
#define WFS_CIM_TYPEREPCONTAINER (3)
#define WFS_CIM_TYPERETRACTCASSETTE (4)
#define WFS_CIM_TYPEREJECT (5)
#define WFS_CIM_TYPECDMSPECIFIC (6)
/* values of WFSCIMCASHIN.fwItemType */
/* values of WFSCIMCASHINTYPE.dwType */
#define WFS_CIM_CITYPALL (0x0001)
#define WFS_CIM_CITYPUNFIT (0x0002)
#define WFS_CIM_CITYPINDIVIDUAL (0x0004)
#define WFS_CIM_CITYPLEVEL3 (0x0008)
#define WFS_CIM_CITYPLEVEL2 (0x0010)
#define WFS_CIM_CITYPIPM (0x0020)

/* 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_STATCUMISSING (4)
#define WFS_CIM_STATCUINOP (5)
#define WFS_CIM_STATCUMISSING (6)
#define WFS_CIM_STATCUMISSING (7)
#define WFS_CIM_STATCUMISSING (8) /* NOTE: Not used in CIM */
#define WFS_CIM_STATCUMANIP (9)

/* values of WFSCIMSTATUS.fwPositions */
/* values of WFSCIMCAPS.fwPositions */
/* values of WFSCIMINPOS.fwPosition */
/* values of WFSCIMTELLERDETAILS.fwInputPosition */
/* values of WFSCIMCASHINSTART.fwInputPosition */
#define WFS_CIM_POSNULL (0x0000)
#define WFS_CIM_POSINLEFT (0x0001)
#define WFS_CIM_POSINRIGHT (0x0002)
#define WFS_CIM_POSINCENTER (0x0004)
#define WFS_CIM_POSINTOP (0x0008)
#define WFS_CIM_POSINBOTTOM (0x0010)
#define WFS_CIM_POSINFRONT (0x0020)
#define WFS_CIM_POSINREAR (0x0040)

/* values of WFSCIMSTATUS.fwPositions */
/* values of WFSCIMCAPS.fwPositions */
/* values of WFSCIMTELLERDETAILS.fwOutputPosition */
/* values of WFSCIMCASHINSTART.fwOutputPosition */
/* values of WFSCIMOUTPUT.fwPosition */
#define WFS_CIM_POSOUTLEFT (0x0080)
#define WFS_CIM_POSOUTRIGHT (0x0100)
#define WFS_CIM_POSOUTCENTER (0x0200)
#define WFS_CIM_POSOUTTOP (0x0400)
#define WFS_CIM_POSOUTBOTTOM (0x0800)
#define WFS_CIM_POSOUTFRONT (0x1000)
#define WFS_CIM_POSOUTREAR (0x2000)

/* values of WFSCIMCASHINSTATUS.wStatus */
#define WFS_CIM_CI_OK (0)
#define WFS_CIM_CIROLLBACK (1)
#define WFS_CIM_CI_ACTIVE (2)
#define WFS_CIM_CI_RETRACT (3)
#define WFS_CIM_CI_UNKNOWN (4)
#define WFS_CIM_CI_RESET (5)

/* values of WFSCIMCAPS.fwRetractAreas */
/* values of WFSCIMRETRACT.usRetractArea */
#define WFS_CIM_RA_RETRACT (0x0001)
#define WFS_CIM_RA_TRANSPORT (0x0002)
#define WFS_CIM_RA_STACKER (0x0004)
#define WFS_CIM_RA_BILLCASSETTES (0x0008)
#define WFS_CIM_RA_NOTSUPP (0x0010)
#define WFS_CIM_RA_REJECT (0x0020)
/* values of WFSCIMP6INFO.usLevel */
/* values of WFSCIMP6SIGNATURE.usLevel */
#define WFS_CIM_LEVEL_2 (2)
#define WFS_CIM_LEVEL_3 (3)
#define WFS_CIM_LEVEL_4 (4)

/* values of WFSCIMTELLERUPDATE.usAction */
#define WFS_CIM_CREATE_TELLER (1)
#define WFS_CIM_MODIFY_TELLER (2)
#define WFS_CIM_DELETE_TELLER (3)

/* values of WFSCIMCUERROR.wFailure */
#define WFS_CIM_CASHUNITEMPTY (1)
#define WFS_CIM_CASHUNITERROR (2)
#define WFS_CIM_CASHUNITFULL (3)
#define WFS_CIM_CASHUNITNOTCONF (4)
#define WFS_CIM_CASHUNITINVALID (5)
#define WFS_CIM_CASHUNITCONFIG (6)
#define WFS_CIM_FEEDMODULEPROBLEM (7)
#define WFS_CIM_CASHUNITPHYSICALLOCKED (8)
#define WFS_CIM_CASHUNITPHYSICALUNLOCKED (9)
#define WFS_CIM_CASHUNITPHYSICALUNLOCKED (10)

/*values of WFSCIMP6SIGNATURE.dwOrientation*/
#define WFS_CIM_ORFRONTTOP (1)
#define WFS_CIM_ORFRONTBOTTOM (2)
#define WFS_CIM_ORBACKTOP (3)
#define WFS_CIM_ORBACKBOTTOM (4)
#define WFS_CIM_ORUNKNOWN (5)
#define WFS_CIM_ORNOTSUPPORTED (6)

/* values for WFSCIMGETITEMINFO.wItemInfoType */
#define WFS_CIM_ITEM_SERIALNUMBER (0x00000001)
#define WFS_CIM_ITEM_SIGNAURE (0x00000002)

/* values of lpusReason in WFS_EXEE_CIM_INPUTREFUSE */
#define WFS_CIM_CASHUNITFULL (1)
#define WFS_CIM_INVALIDBILL (2)
#define WFS_CIM_NOBILLSTODEPOSIT (3)
#define WFS_CIM_DEPOSITFAILURE (4)
#define WFS_CIM_COMMINPCOMPFAILURE (5)
#define WFS_CIM_STACKERFULL (6)
#define WFS_CIM_FOREIGN_ITEMS_DETECTED (7)
#define WFS_CIM_INVALIDBUNCH (8)
#define WFS_CIM_COUNTERFEIT (9)
#define WFS_CIM_LIMITOVERTOTALITEMS (10)
#define WFS_CIM_LIMITOVERAMOUNT (11)

/* values of lpusReason in WFS_EXEE_CIM_NOTESERROR */
#define WFS_CIM_DOUBLENOTEDETECTED (1)
#define WFS_CIM_LONGNOTEDETECTED (2)
#define WFS_CIM_SKewedNOTE (3)
#define WFS_CIM_INCORRECTCOUNT (4)
#define WFS_CIM_NOTESTOOCLOSE (5)
#define WFS_CIM_OTHERNOTEERROR (6)
#define WFS_CIM_SHORTNOTEDETECTED (7)

/* Values of fwUsage in WFS_INF_CIM_POSITION_CAPABILITIES */
#define WFS_CIM_POSIN (0x0001)
#define WFS_CIM_POSREFUSE (0x0002)
#define WFS_CIM_POSROLLBACK (0x0004)
/* values of WFSCIMPOSITIONINFO.wAdditionalBunches */
#define WFS_CIM_ADDBUNCHNONE (1)
#define WFS_CIM_ADDBUNCHONEMORE (2)
#define WFS_CIM_ADDBUNCHUNKNOWN (3)
/* values of WFSCIMPOSITIONINFO.usBunchesRemaining */
#define WFS_CIM_NUMBERUNKNOWN (255)
/* values of WFSCIMCAPS.fwCountActions */
#define WFS_CIM_COUNTNOTSUPP (0x0000)
#define WFS_CIM_COUNTINDIVIDUAL (0x0001)
#define WFS_CIM_COUNTALL (0x0002)
/* values of WFSCIMDEVICELOCKCONTROL.wDeviceAction */
/* values of WFSCIMDEVICELOCKCONTROL.wCashUnitAction */
/* values of WFSCIMUNITLOCKCONTROL.wUnitAction */
#define WFS_CIM_LOCK (1)
#define WFS_CIM_UNLOCK (2)
#define WFS_CIM_LOCKALL (3)
#define WFS_CIM_UNLOCKALL (4)
#define WFS_CIM_LOCKINDIVIDUAL (5)
#define WFS_CIM_NOLOCKACTION (6)
#define WFS_CIM_LOCKUNKNOWN (7)
#define WFS_CIM_LOCKNOTSUPPORTED (8)
/* values of WFSCIMSTATUS.wAntiFraudModule */
#define WFS_CIM_AFMNOTSUPP (0)
#define WFS_CIM_AFMOK (1)
#define WFS_CIM_AFMINOP (2)
#define WFS_CIM_AFMDEVICEDETECTED (3)
#define WFS_CIM_AFMUNKNOWN (4)
/* 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_ITEMSTAKEN (-CIM_SERVICE_OFFSET + 15))
#define WFS_ERR_CIM_CASHINACTIVE (-CIM_SERVICE_OFFSET + 17))
#define WFS_ERR_CIM_EXCHANGEACTIVE (-CIM_SERVICE_OFFSET + 19))
#define WFS_ERR_CIM_MEDIAINACTIVE  
(-(CIM_SERVICE_OFFSET + 47))

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

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

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

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

typedef struct _wfs_cim_physicalcu
{
  LPSTR                 lpPhysicalPositionName;
  CHAR                   cUnitID[5];
}
typedef struct _wfs_cim_note_number
{
    USHORT usNoteID;
    ULONG ulCount;
} WFSCIMNOTENUMBER, *LPWFSCIMNOTENUMBER;

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

typedef struct _wfs_cim_cash_in
{
    USHORT usNumber;
    DWORD fwType;
    DWORD fwItemType;
    CHAR cUnitID[5];
    CHAR cCurrencyID[3];
    ULONG ulValues;
    ULONG ulCashInCount;
    ULONG ulCount;
    ULONG ulMaximum;
    USHORT usStatus;
    BOOL bAppLock;
    LPWFSCIMNOTENUMBERLIST lpNoteNumberList;
    USHORT usNumPhysicalCUs;
    LPWFSCIMPHCU *lppPhysical;
    LPSTR lpszExtra;
    LPUSHORT lpusNoteIDs;
    WORD usCDMType;
    LPSTR lpszCashUnitName;
    ULONG ulInitialCount;
    ULONG ulDispensedCount;
    ULONG ulPresentedCount;
    ULONG ulRetractedCount;
    ULONG ulRejectCount;
    ULONG ulMinimum;
} WFSCIMCASHIN, *LPWFSCIMCASHIN;

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

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

typedef struct _wfs_cim_teller_totals
{
    CHAR cCurrencyID[3];
    ULONG ulItemsReceived;
    ULONG ulItemsDispensed;
}
typedef struct _wfs_cim_teller_details
{
    USHORT  usTellerID;
    WORD    fwInputPosition;
    WORD    fwOutputPosition;
    LPWFS CIMTELLERTOTALS  *lppTellerTotals;
} WFSCIMTELLERDETAILS, *LPWFSCIMTELLERDETAILS;

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

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

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

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

typedef struct _wfs_cim_P6_info
{
    USHORT  usLevel;
    LPWFSCIMNOTENUMBERLIST  lpNoteNumberList;
    USHORT  usNumOfSignatures;
} WFSCIMP6INFO, *LPWFSCIMP6INFO;

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

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

typedef struct _wfs_cim_get_item_info
{
    USHORT  usLevel;
    USHORT  usIndex;
    DWORD   dwItemInfoType;
}
typedef struct _wfs_cim_item_info
{
    USHORT usNoteID;
    LPWSTR lpszSerialNumber;
    LPWFSCIMP6SIGNATURE lpP6Signature;
} WFSCIMITEMINFO, *LPWFSCIMITEMINFO;

typedef struct _wfs_cim_item_info_summary
{
    USHORT usLevel;
    USHORT usNumOfItems;
} WFSCIMITEMINFOSUMMARY, *LPWFSCIMITEMINFOSUMMARY;

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

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

typedef struct _wfs_cim_replenish_info
{
    USHORT usNumberSource;
} WFSCIMREPiNFO, *LPWFSCIMREPiNFO;

typedef struct _wfs_cim_replenish_info_target
{
    USHORT usNumberTarget;
} WFSCIMREPiNFOTARGET, *LPWFSCIMREPiNFOTARGET;

typedef struct _wfs_cim_replenish_info_result
{
    LPWFSCIMREPiNFOTARGET *lppReplenishTargets;
} WFSCIMREPiNFORES, *LPWFSCIMREPiNFORES;

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

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

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

typedef struct _wfs_cim_cash_unit_capabilities
{
    USHORT usNumber;
}
USHORT                   usNumPhysicalCUs;
LPWFSCLPHCUCAPABILITIES *lppPhysical;
BOOL                     bRetractNoteCountThresholds;
LPSTR                    lpszExtra;
} WFSCIMCASHUNTCAPABILITIES, *LPWFSCIMCASHUNTCAPABILITIES;

typedef struct _wfs_cim_cash_caps
    
    USHORT                   usCount;
    LPWFSICASHCAPABILITIES   *lppCashUnitCaps;
} WFSCIMCASHCAPABILITIES, *LPWFSCIMCASHCAPABILITIES;

/*=================================================================*
/* 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;
    LPWFSICIMTELLERDETAILS   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;
    LPUUSHORT                lpusCUNumList;
    LPWFSICIMOUTPUT          lpOutput;
} WFSCIMSTARTEX, *LPWFSCIMSTARTEX;

typedef struct _wfs_cim_itemposition
    
    USHORT                   usNumber;
    LPWFSIMRETRACT           lpRetractArea;
    WORD                     fwOutputPosition;
} WFSCIMITEMPOSITION, *LPWFSCIMITEMPOSITION;

typedef struct _wfs_cim_cash_in_type
    
    USHORT                   usNumber;
    DWORD                    dwType;
    LPUUSHORT                lpusNoteIDs;
} WFSCIMCASHINTYPE, *LPWFSCIMCASHINTYPE;

typedef struct _wfs_cim_set_guidlight
    
    WORD                     wGuidLight;
typedef struct _wfs_cim_configure_note_reader
{
    BOOL bLoadAlways;
} WFSCIMCONFIGURENOTERECORDER, *LPWFSCIMCONFIGURENOTERECORDER;

datatype struct _wfs_cim_configure_note_reader_out
{
    BOOL bRebootNecessary;
} WFSCIMCONFIGURENOTERECORDEROUT, *LPWFSCIMCONFIGURENOTERECORDEROUT;

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

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

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

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

typedef struct _wfs_cim_replenish_target
{
    USHORT usNumberTarget;
    ULONG ulNumberOfItemsToMove;
    BOOL bRemoveAll;
} WFSCIMREPTARGET, *LPWFSCIMREPTARGET;

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

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

typedef struct _wfs_cim_replenish_result
{
    ULONG ulNumberOfItemsRemoved;
    ULONG ulNumberOfItemsRejected;
    LPWFSCIMREPTARGETRES *lppReplenishTargetResults;
} WFSCIMREPRRES, *LPWFSCIMREPRRES;

typedef struct _wfs_cim_amount_limit
{
    CHAR cCurrencyID[3];
    ULONG ulAmount;
} WFSCIMAMOUNTLIMIT, *LPWFSCIMAMOUNTLIMIT;
typedef struct _wfs_cim_cash_in_limit
{
    ULONG ulTotalItemsLimit;
    LPWFSCIMAMOUNTLIMIT lpAmountLimit;
} WFSCIMCASHINLIMIT, *LPWFSCIMCASHINLIMIT;

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

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

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

typedef struct _wfs_cim_setmode
{
    WORD wMixedMode;
} WFSCIMSETMODE, *LPWFSCIMSETMODE;

typedef struct _wfs_cim_present
{
    WORD fwPosition;
} WFSCIMPRESENT, *LPWFSCIMPRESENT;

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

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

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

typedef struct _wfs_cim_position_info
{
    WORD wPosition;
    WORD wAdditionalBunches;
    USHORT usBunchesRemaining;
} WFSCIMPOSITIONINFO, *LPWFSCIMPOSITIONINFO;

typedef struct _wfs_cim_device_position
{
    WORD wPosition;
} WFSCIMDEVICEPOSITION, *LPWFSCIMDEVICEPOSITION;

typedef struct _wfs_cim_power_save_change
{
    USHORT usPowerSaveRecoveryTime;
} WFSCIMPOWERSAVECHANGE, *LPWFSCIMPOWERSAVECHANGE;

typedef struct _wfs_cim_incomplete_replenish
{ LPWFSCIMREPREPRES lpReplenish; } WFSCIMINCOMPLETEREPLENISH, *LPWFSCIMINCOMPLETEREPLENISH;

/* restore alignment */
#pragma pack (pop)
#endif /* __cplusplus */
#endif /* __INC_XFSCIM_H */