Extensions for Financial Services (XFS) interface specification
Release 3.20 - Part 12: Camera 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 ....................................................................................................................... 3  

1. Introduction .......................................................................................................... 6  
   1.1 Background to Release 3.20 ................................................................................. 6  
   1.2 XFS Service-Specific Programming ................................................................ 6  

2. Banking Cameras ................................................................................................ 7  

3. References ........................................................................................................... 8  

4. Info Commands ................................................................................................ 9  
   4.1 WFS_INF_CAM_STATUS ............................................................................... 9  
   4.2 WFS_INF_CAM_CAPABILITIES ................................................................. 12  

5. Execute Commands ......................................................................................... 14  
   5.1 WFS_CMD_CAM_TAKE_PICTURE ............................................................... 14  
   5.2 WFS_CMD_CAM_RESET .......................................................................... 15  
   5.3 WFS_CMD_CAM_TAKE_PICTURE_EX ...................................................... 16  

6. Events ............................................................................................................... 18  
   6.1 WFS_USRE_CAM_MEDIATHRESHOLD ....................................................... 18  
   6.2 WFS_EXEE_CAM_INVALIDDATA .............................................................. 19  

7. C - Header file .................................................................................................. 20
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
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 (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 (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>3.10</td>
<td>November 29, 2007</td>
<td>For a description of changes from version 3.00 to version 3.10 see the CAM 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 CAM 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 3.10 of the XFS specification is based on a C API and is delivered with the continued promise for the protection of technical investment for existing applications. This release of the XFS specification has been prompted by a series of factors.

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

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

1.2 XFS Service-Specific Programming

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

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

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

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

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

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

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

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

This specification describes the functionality of the services provided by the Camera (CAM) services under XFS, by defining the service-specific commands that can be issued, using the \texttt{WFSGetInfo}, \texttt{WFSAsyncGetInfo}, \texttt{WFSEexecute} and \texttt{WFSAasyncExecute} functions.

Banking camera systems usually consist of a recorder, a video mixer and one or more cameras. If there are several cameras, each camera focuses a special place within the self-service area (e.g. the room, the customer or the cash tray). By using the video mixer it can be decided, which of the cameras should take the next photo. Furthermore data can be given to be inserted in the photo (e.g. date, time or bank code).

If there is only one camera that can switch to take photos from different positions, it is presented by the Service Provider as a set of cameras, one for each of its possible positions.
3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer’s Reference Revision 3.20
4. Info Commands

4.1 WFS_INF_CAM_STATUS

Description
This command reports the full range of information available, including the information that is provided by the Service Provider.

Input Param
None.

Output Param
LPWFSCAMSTATUS lpStatus;

typedef struct _wfs_cam_status
{
    WORD    fwDevice;
    WORD    fwMedia[WFS_CAM_CAMERAS_SIZE];
    WORD    fwCameras[WFS_CAM_CAMERAS_SIZE];
    USHORT   usPictures[WFS_CAM_CAMERAS_SIZE];
    LPSTR    lpszExtra;
    WORD    wAntiFraudModule;
} WFSCAMSTATUS, *LPWFSCAMSTATUS;

fwDevice
Specifies the state of the Camera device as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_DEVONLINE</td>
<td>The device is online (i.e. powered on and operable).</td>
</tr>
<tr>
<td>WFS_CAM_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_CAM_DEVPOWEROFF</td>
<td>The device is powered off or physically not connected.</td>
</tr>
<tr>
<td>WFS_CAM_DEVNODEVICE</td>
<td>There is no device 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_CAM_DEVHWERROR</td>
<td>The device is inoperable due to a hardware error.</td>
</tr>
<tr>
<td>WFS_CAM_DEVUSERERROR</td>
<td>The device is inoperable because a person is preventing proper operation.</td>
</tr>
<tr>
<td>WFS_CAM_DEVBUSY</td>
<td>The device is busy and not able to process an execute command at this time.</td>
</tr>
<tr>
<td>WFS_CAM_DEVFRAUDATTEMPT</td>
<td>The device is present but is inoperable because it has detected a fraud attempt.</td>
</tr>
<tr>
<td>WFS_CAM_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>

fwMedia […]
Specifies the state of the recording media of the cameras. A number of indexes are defined below.
The maximum fwMedia index is WFS_CAM_CAMERAS_MAX.

fwMedia [WFS_CAM_ROOM]
Specifies the state of the recording media of the camera that monitors the whole self-service area.
Specified as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_MEDIAOK</td>
<td>The media is in a good state.</td>
</tr>
<tr>
<td>WFS_CAM.MEDIAHIGH</td>
<td>The media is almost full (threshold).</td>
</tr>
<tr>
<td>WFS_CAM.MEDIAFULL</td>
<td>The media is full.</td>
</tr>
<tr>
<td>WFS_CAM_MEDIANOTSUPP</td>
<td>The device does not support sensing the media level.</td>
</tr>
</tbody>
</table>
### fwMedia [WFS_CAM_PERSON]

Specifies the state of the recording media of the camera that monitors the person standing in front of the self-service machine. Specified as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_MEDIAOK</td>
<td>The media is in a good state.</td>
</tr>
<tr>
<td>WFS_CAM_MEDIAHIGH</td>
<td>The media is almost full (threshold).</td>
</tr>
<tr>
<td>WFS_CAM_MEDIAFULL</td>
<td>The media is full.</td>
</tr>
<tr>
<td>WFS_CAM_MEDIANOTSUPP</td>
<td>The device does not support sensing the media level.</td>
</tr>
<tr>
<td>WFS_CAM.MEDIAUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the media cannot be determined.</td>
</tr>
</tbody>
</table>

### fwMedia [WFS_CAM_EXITSLot]

Specifies the state of the recording media of the camera that monitors the exit slot(s) of the self-service machine. Specified as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_MEDIAOK</td>
<td>The media is in a good state.</td>
</tr>
<tr>
<td>WFS_CAM_MEDIAHIGH</td>
<td>The media is almost full (threshold).</td>
</tr>
<tr>
<td>WFS_CAM_MEDIAFULL</td>
<td>The media is full.</td>
</tr>
<tr>
<td>WFS_CAM_MEDIANOTSUPP</td>
<td>The device does not support sensing the media level.</td>
</tr>
<tr>
<td>WFS_CAM.MEDIAUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the media cannot be determined.</td>
</tr>
</tbody>
</table>

### fwCameras [WFS_CAM_CAMERAS_MAX]

Specifies the state of the cameras. A number of cameras are defined below. The maximum camera index is WFS_CAM_CAMERAS_MAX.

### fwCameras [WFS_CAM_PERSON]

Specifies the state of the camera that monitors the person standing in front of the self-service machine. Specified as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_CAMNOTSUPP</td>
<td>The camera is not supported.</td>
</tr>
<tr>
<td>WFS_CAM_CAMOK</td>
<td>The camera is in a good state.</td>
</tr>
<tr>
<td>WFS_CAM_CAMINOP</td>
<td>The camera is inoperative.</td>
</tr>
<tr>
<td>WFS_CAM_CAMUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the camera cannot be determined.</td>
</tr>
</tbody>
</table>

### fwCameras [WFS_CAM_ROOM]

Specifies the state of the camera that monitors the whole self-service area. Specified as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_CAMNOTSUPP</td>
<td>The camera is not supported.</td>
</tr>
<tr>
<td>WFS_CAM_CAMOK</td>
<td>The camera is in a good state.</td>
</tr>
<tr>
<td>WFS_CAM_CAMINOP</td>
<td>The camera is inoperative.</td>
</tr>
<tr>
<td>WFS_CAM_CAMUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the camera cannot be determined.</td>
</tr>
</tbody>
</table>

### fwCameras [WFS_CAM_EXITSLot]

Specifies the state of the camera that monitors the exit slot(s) of the self-service machine. Specified as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_CAMNOTSUPP</td>
<td>The camera is not supported.</td>
</tr>
<tr>
<td>WFS_CAM_CAMOK</td>
<td>The camera is in a good state.</td>
</tr>
<tr>
<td>WFS_CAM_CAMINOP</td>
<td>The camera is inoperative.</td>
</tr>
<tr>
<td>WFS_CAM_CAMUNKNOWN</td>
<td>Due to a hardware error or other condition, the state of the camera cannot be determined.</td>
</tr>
</tbody>
</table>
usPictures [...]  
Specifies the number of pictures stored on the recording media of the cameras.  
A number of indexes are defined below. The maximum usPictures index is WFS_CAM_CAMERAS_MAX.

<table>
<thead>
<tr>
<th>Index</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_ROOM</td>
<td>The camera that monitors the whole self-service area.</td>
</tr>
<tr>
<td>WFS_CAM_PERSON</td>
<td>The camera that monitors the person standing in front of the self-service machine.</td>
</tr>
<tr>
<td>WFS_CAM_EXITSLOT</td>
<td>The camera that monitors the exit slot(s) of the self-service machine.</td>
</tr>
</tbody>
</table>

lpszExtra  
Point 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.

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_CAM_AFMNOTSUPP</td>
<td>No anti-fraud module is available.</td>
</tr>
<tr>
<td>WFS_CAM_AFMOK</td>
<td>Anti-fraud module is in a good state and no foreign device is detected.</td>
</tr>
<tr>
<td>WFS_CAM_AFMINOP</td>
<td>Anti-fraud module is inoperable.</td>
</tr>
<tr>
<td>WFS_CAM_AFMDEVICEDETECTED</td>
<td>Anti-fraud module detected the presence of a foreign device.</td>
</tr>
<tr>
<td>WFS_CAM_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 require or expect specific information to be present in the lpszExtra parameter may not be device or vendor-independent.

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

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

Description
This command is used to retrieve the capabilities of the camera system.

Input Param
None.

Output Param LPWFSCAMCAPS lpCaps;

```c
typedef struct _wfs_cam_caps
{
    WORD    wClass;
    WORD    fwType;
    WORD    fwCameras[WFS_CAM_CAMERAS_SIZE];
    USHORT   usMaxPictures;
    WORD    fwCamData;
    USHORT   usMaxDataLength;
    WORD    fwCharSupport;
    LPSTR    lpszExtra;
    BOOL    bPictureFile;
    BOOL    bAntiFraudModule;
} WFSCAMCAPS, *LPWFSCAMCAPS;
```

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

**fwType**
Specifies the type of the camera device; only current value is:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_TYPE_CAM</td>
<td>Camera system.</td>
</tr>
</tbody>
</table>

**fwCameras [...]**
Specifies which cameras are available. A number of cameras are defined below. The maximum camera index is WFS_CAM_CAMERAS_MAX.

**fwCameras [WFS_CAM_ROOM]**
Specifies whether the camera that monitors the whole self-service area is available. Specified as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_NOT_AVAILABLE</td>
<td>This camera is not available.</td>
</tr>
<tr>
<td>WFS_CAM_AVAILABLE</td>
<td>This camera is available.</td>
</tr>
</tbody>
</table>

**fwCameras [WFS_CAM_PERSON]**
Specifies whether the camera that monitors the person standing in front of the self-service machine is available. Specified as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_NOT_AVAILABLE</td>
<td>This camera is not available.</td>
</tr>
<tr>
<td>WFS_CAM_AVAILABLE</td>
<td>This camera is available.</td>
</tr>
</tbody>
</table>

**fwCameras [WFS_CAM_EXIT SLOT]**
Specifies whether the camera that monitors the exit slot(s) of the self-service machine is available. Specified as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_NOT_AVAILABLE</td>
<td>This camera is not available.</td>
</tr>
<tr>
<td>WFS_CAM_AVAILABLE</td>
<td>This camera is available.</td>
</tr>
</tbody>
</table>

**usMaxPictures**
Specifies the maximum number of pictures that can be stored on the recording media.

**fwCamData**
Specifies, if data can be added to the picture. Specified as a combination of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_NOTADD</td>
<td>No data can be added to the picture.</td>
</tr>
<tr>
<td>WFS_CAM_AUTOADD</td>
<td>Data is added automatically to the picture.</td>
</tr>
</tbody>
</table>
WFS_CAM_MANADD  Data can be added manually to the picture using the field lpszCamData in the WFS_CMD_CAM_TAKE_PICTURE command.

`usMaxDataLength`
Specifies the maximum length of the data that is displayed on the photo. Zero, if data cannot be manually added to the picture.

`fwCharSupport`
One or more flags specifying the Character Set supported by the Service Provider:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_ASCII</td>
<td>ASCII is supported for execute command data values.</td>
</tr>
<tr>
<td>WFS_CAM_UNICODE</td>
<td>UNICODE is supported for execute command data values.</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.

`bPictureFile`
Specifies whether the WFS_CMD_CAM_TAKE_PICTURE_EX command, which enables applications to specify the file path and name of a picture to be taken, is supported.

`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 require or expect specific information to be present in the `lpszExtra` parameter may not be device or vendor-independent.
5. Execute Commands

5.1 WFS_CMD_CAM_TAKE_PICTURE

Description
This command is used to start the recording of the camera system. It is possible to select which camera or which camera position should be used to take a picture. Data to be displayed on the photo can be specified using the \textit{lpszCamData} or \textit{lpszUNICODECamData} parameter.

Input Param
LPWFSCAMTAKEPICT lpTakePict;

typedef struct _wfs_cam_take_picture
{
  WORD    wCamera;
  LPSTR    lpszCamData;
  LPWSTR   lpszUNICODECamData;
} WFSCAMTAKEPICT, *LPWFSCAMTAKEPICT;

\textit{wCamera}
Specifies the camera that should take the photo as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_ROOM</td>
<td>Monitors the whole self-service area.</td>
</tr>
<tr>
<td>WFS_CAM_PERSON</td>
<td>Monitors the person standing in front of the self-service machine.</td>
</tr>
<tr>
<td>WFS_CAM_EXITSLOT</td>
<td>Monitors the exit slot(s) of the self-service machine.</td>
</tr>
</tbody>
</table>

\textit{lpszCamData}
Specifies the text string to be displayed on the photo. If the maximum text length is exceeded it will be truncated. In this case or if the text given is invalid an execute event WFS_EXEE_CAM_INVALIDDATA is generated. Nevertheless the picture is taken.

\textit{lpszUNICODECamData}
Specifies the UNICODE text string to be displayed on the photo. If the maximum text length is exceeded, it will be truncated. In this case or if the text given is invalid an execute event WFS_EXEE_CAM_INVALIDDATA is generated. Nevertheless the picture is taken.

The \textit{lpszUNICODECamData} field should only be used if the Service Provider supports UNICODE. The \textit{lpszCamData} and \textit{lpszUNICODECamData} fields are mutually exclusive.

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_CAM_CAMNOTSUPP</td>
<td>The specified camera is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CAM_MEDIAFULL</td>
<td>The recording media is full.</td>
</tr>
<tr>
<td>WFS_ERR_CAM_CAMINOP</td>
<td>The specified camera is inoperable.</td>
</tr>
<tr>
<td>WFS_ERR_CAM_CHARSETNOTSUPP</td>
<td>Character set(s) supported by Service Provider is inconsistent with use of \textit{lpszCamData} or \textit{lpszUNICODECamData} fields.</td>
</tr>
<tr>
<td>WFS_ERR_CAM_FILEIOERROR</td>
<td>Directory does not exist or File IO error while storing the image to the hard disk.</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_CAM_MEDIATHRESHOLD</td>
<td>The state of the recording media reached a threshold.</td>
</tr>
<tr>
<td>WFS_EXEE_CAM_INVALIDDATA</td>
<td>The text string given is too long or in some other way invalid.</td>
</tr>
</tbody>
</table>

Comments
None.
5.2 WFS_CMD_CAM_RESET

<table>
<thead>
<tr>
<th>Description</th>
<th>Sends a service reset to the Service Provider.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Param</td>
<td>None.</td>
</tr>
<tr>
<td>Output Param</td>
<td>None.</td>
</tr>
<tr>
<td>Error Codes</td>
<td>Only the generic error codes defined in [Ref. 1] can be generated by this command.</td>
</tr>
<tr>
<td>Events</td>
<td>Only the generic events defined in [Ref. 1] can be generated by this command.</td>
</tr>
<tr>
<td>Comments</td>
<td>This command is used by an application control program to cause a device to reset itself to a known good condition.</td>
</tr>
</tbody>
</table>
5.3 WFS_CMD_CAM_TAKE_PICTURE_EX

Description
This command is used to start the recording of the camera system. It is possible to select which
camera or which camera position should be used to take a picture. Data to be displayed on the
photo can be specified using the lpszCamData or lpszUNICODECamData parameter.

Input Param
LPWFS_CUSTOM TakePictEx;

typedef struct _wfs_cam_take_picture_ex
{
    WORD    wCamera;
    LPSTR    lpszCamData;
    LPWSTR   lpszUNICODECamData;
    LPSTR    lpszPictureFile;
} WFS_CUSTOM_TAKEPICTEX, *LPWFS_CUSTOM_TAKEPICTEX;

wCamera
Specifies the camera that should take the photo as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_ROOM</td>
<td>Monitors the whole self-service area.</td>
</tr>
<tr>
<td>WFS_CAM_PERSON</td>
<td>Monitors the person standing in front of the self-service machine.</td>
</tr>
<tr>
<td>WFS_CAM_EXITSLT</td>
<td>Monitors the exit slot(s) of the self-service machine.</td>
</tr>
</tbody>
</table>

lpszCamData
Specifies the text string to be displayed on the photo. If the maximum text length is exceeded it will be truncated. In this case or if the text given is invalid an execute event WFS_EXE_IE_CAM_INVALIDDATA is generated. Nevertheless the picture is taken.

lpszUNICODECamData
Specifies the UNICODE text string to be displayed on the photo. If the maximum text length is exceeded, it will be truncated. In this case or if the text given is invalid an execute event WFS_EXE_IE_CAM_INVALIDDATA is generated. Nevertheless the picture is taken.

The lpszUNICODECamData field should only be used if the Service Provider supports UNICODE. The lpszCamData and lpszUNICODECamData fields are mutually exclusive.

lpszPictureFile
Specifies the full path and file name of the image to be taken by a camera device. The file name includes the image format specific file extension. The Service Provider is responsible for converting the image into the required format.

This value is terminated with a single null character and cannot contain UNICODE characters.

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_CAM_CAMNOTSUPP</td>
<td>The specified camera is not supported.</td>
</tr>
<tr>
<td>WFS_ERR_CAM_MEDIAFULL</td>
<td>The recording media is full.</td>
</tr>
<tr>
<td>WFS_ERR_CAM_CAMINOP</td>
<td>The specified camera is inoperable.</td>
</tr>
<tr>
<td>WFS_ERR_CAM_CHARSETNOTSUPP</td>
<td>Character set(s) supported by Service Provider is inconsistent with use of lpszCamData or lpszUNICODECamData fields.</td>
</tr>
<tr>
<td>WFS_ERR_CAM_FILEIOERROR</td>
<td>Directory does not exist or File IO error while storing the image to the hard disk.</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_CAM_MEDIATHRESHOLD</td>
<td>The state of the recording media reached a threshold.</td>
</tr>
</tbody>
</table>
WFS_EXEE_CAM_INVALIDDATA

Comments
None.

The text string given is too long or in some other way invalid.
6. Events

6.1 WFS_USRE_CAM_MEDIATRESHOLD

**Description**
This user event is used to specify that the state of the recording media reached a threshold.

**Event Param**
LPWORD lpwMediaThreshold;

*lpwMediaThreshold*
Specified as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_CAM_MEDIAOK</td>
<td>The recording media is a good state.</td>
</tr>
<tr>
<td>WFS_CAM_MEDIAHIGH</td>
<td>The recording media is almost full.</td>
</tr>
<tr>
<td>WFS_CAM_MEDIAFULL</td>
<td>The recording media is full.</td>
</tr>
</tbody>
</table>

**Comments**
None.
### 6.2 WFS_EXEE_CAM_INVALIDDATA

<table>
<thead>
<tr>
<th>Description</th>
<th>This execute event is used to specify that the text string given was too long or in some other way invalid.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Event Param</td>
<td>None.</td>
</tr>
<tr>
<td>Comments</td>
<td>None.</td>
</tr>
</tbody>
</table>
7. C - Header file

/* ***************************************************************************/
/* XFS - Camera (CAM) definitions */
/* Version 3.20  (March 02 2011) */
/* **************************************************************************/ay

#ifndef __INC_XFSCAM__H
#define __INC_XFSCAM__H
#ifdef __cplusplus
extern "C" {
#include <xfsapi.h>
/* be aware of alignment */
#pragma pack (push, 1)
/* values of WFSCAMCAPS.wClass */
#define WFS_SERVICE_CLASS_CAM               (10)
#define WFS_SERVICE_VERSION_CAM             (0x1403) /* Version 3.20 */
#define WFS_SERVICE_NAME_CAM                "CAM"
#define CAM_SERVICE_OFFSET                  (WFS_SERVICE_CLASS_CAM * 100)
/* CAM Info Commands */
#define WFS_INF_CAM_STATUS                  (CAM_SERVICE_OFFSET + 1)
#define WFS_INF_CAM_CAPABILITIES            (CAM_SERVICE_OFFSET + 2)
/* CAM Execute Commands */
#define WFS_CMD_CAM_TAKE_PICTURE            (CAM_SERVICE_OFFSET + 1)
#define WFS_CMD_CAM_RESET                   (CAM_SERVICE_OFFSET + 2)
#define WFS_CMD_CAM_TAKE_PICTURE_EX         (CAM_SERVICE_OFFSET + 3)
/* CAM Messages */
#define WFS_USRE_CAM_MEDIATHRESHOLD         (CAM_SERVICE_OFFSET + 1)
#define WFS_EXEE_CAM_INVALIDDATA            (CAM_SERVICE_OFFSET + 2)
/* values of WFSCAMSTATUS.fwDevice */
#define WFS_CAM_DEVONLINE                   WFS_STAT_DEVONLINE
#define WFS_CAM_DEVOPOFFLINE                WFS_STAT_DEVOPOFFLINE
#define WFS_CAM_DEVPOWEROFF                 WFS_STAT_DEVPOWEROFF
#define WFS_CAM_DEVDENODEVICE               WFS_STAT_DEVDENODEVICE
#define WFS_CAM_DEVHERROR                    WFS_STAT_DEVHERROR
#define WFS_CAM_DEVUSERERROR                 WFS_STAT_DEVUSERERROR
#define WFS_CAM_DEVBUSY                      WFS_STAT_DEVBUSY
#define WFS_CAM_DEVFRAUDATTEMPT             WFS_STAT_DEVFRAUDATTEMPT
#define WFS_CAM_DEVPOTENTIALFRAUD           WFS_STAT_DEVPOTENTIALFRAUD
/* number of cameras supported/length of WFSCAMSTATUS.fwCameras field */
#define WFS_CAM_CAMERAS_SIZE                (8)
#define WFS_CAM_CAMERAS_MAX                 (WFS_CAM_CAMERAS_SIZE - 1)
/* indices of WFSCAMSTATUS.fwMedia [...]*/
WFSCAMSTATUS.fwCameras [...] WFS_CAM_CAMERAS_MAX
*/

CWA 16374-12:2011 (E)
#define WFS_CAM_ROOM (0)
#define WFS_CAM_PERSON (1)
#define WFS_CAM_EXITSLOT (2)

/* values of WFSCAMSTATUS.fwMedia */
#define WFS_CAM_MEDIAOK (0)
#define WFS_CAM_MEDIAHIGH (1)
#define WFS_CAM_MEDIAFULL (2)
#define WFS_CAM_MEDIAUNKNOWN (3)
#define WFS_CAM_MEDIANOTSUPP (4)

/* values of WFSCAMSTATUS.fwCameras */
#define WFS_CAM_CAMNOTSUPP (0)
#define WFS_CAM_CAMOK (1)
#define WFS_CAM_CAMINOP (2)
#define WFS_CAM_CAMUNKNOWN (3)

/* values of WFSCAMCAPS.fwType */
#define WFS_CAM_TYPE_CAM (1)

/* values of WFSCAMCAPS.fwCameras */
#define WFS_CAM_NOT_AVAILABLE (0)
#define WFS_CAM_AVAILABLE (1)

/* values of WFSCAMCAPS.fwCamData */
#define WFS_CAM_NOTADD (0)
#define WFS_CAM_AUTOADD (1)
#define WFS_CAM_MANADD (2)

/* values of WFSCAMCAPS.fwCharSupport */
#define WFS_CAM_ASCII (0x0001)
#define WFS_CAM_UNICODE (0x0002)

/* values of WFSCAMSTATUS.wAntiFraudModule */
#define WFS_CAM_AFMNOTSUPP (0)
#define WFS_CAM_AFMOK (1)
#define WFS_CAM_AFMINOP (2)
#define WFS_CAM_AFMDEVICEDETECTED (3)
#define WFS_CAM_AFMUNKNOWN (4)

/* XFS CAM Errors */
#define WFS_ERR_CAM_CAMNOTSUPP (-CAM_SERVICE_OFFSET)
#define WFS_ERR_CAM_CAMINOP (-CAM_SERVICE_OFFSET + 1)
#define WFS_ERR_CAM_CHARSETNOTSUPP (-CAM_SERVICE_OFFSET + 2)
#define WFS_ERR_CAM_FILEIOERROR (-CAM_SERVICE_OFFSET + 3)

typedef struct _wfs_cam_status
{
    WORD      fwDevice;
    WORD      fwMedia[WFS_CAM_CAMERAS_SIZE];
    WORD      fwCameras[WFS_CAM_CAMERAS_SIZE];
    USHORT    usPictures[WFS_CAM_CAMERAS_SIZE];
    LPSTR     lpszExtra;
    WORD      wAntiFraudModule;
} WFSCAMSTATUS, *LPWFSCAMSTATUS;

typedef struct _wfs_cam_caps
typedef struct _wfs_cam_take_picture
{
    WORD wCamera;
    LPSTR lpszCamData;
    LPWSTR lpszUNICODECamData;
} WFSCAMTAKEPICT, *LPWFSCAMTAKEPICT;

typedef struct _wfs_cam_take_picture_ex
{
    WORD wCamera;
    LPSTR lpszCamData;
    LPWSTR lpszUNICODECamData;
    LPSTR lpszPictureFile;
} WFSCAMTAKEPICTEX, *LPWFSCAMTAKEPICTEX;

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

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

#else /*INC_XFSCAM__H */