CEN

WORKSHOP

AGREEMENT

CWA 16926-71

January 2023

ICS 35.200; 35.240.40; 35.240.15

English version

Extensions for Financial Services (XFS) interface specification Release 3.50 - Part 71: Camera Device Class Interface - Programmer's Reference - Migration from Version 3.40 (CWA 16926:2020) to Version 3.50 (this CWA)

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

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

Ref. No.:CWA 16926-71:2023 E

Table of Contents

European Foreword

This CEN Workshop Agreement has been developed in accordance with the CEN-CENELEC Guide 29 "CEN/CENELEC Workshop Agreements – The way to rapid consensus" and with the relevant provisions of CEN/CENELEC Internal Regulations - –art 2. It was approved by a Workshop of representatives of interested parties on 2022-11-08, the constitution of which was supported by CEN following several public calls for participation, the first of which was made on 1998-06-24. However, this CEN Workshop Agreement does not necessarily include all relevant stakeholders.
The final text of this CEN Workshop Agreement was provided to CEN for publication on 2022-11-18.
The following organizations and individuals developed and approved this CEN Workshop Agreement:

- AURIGA SPA
- CIMA SPA
- DIEBOLD NIXDORF SYSTEMS GMBH
- FIS BANKING SOLUTIONS UK LTD (OTS)
- FUJITSU TECHNOLOGY SOLUTIONS
- GLORY LTD
- GRG BANKING EQUIPMENT HK CO LTD
- HITACHI CHANNEL SOLUTIONS CORP
- HYOSUNG TNS INC
- JIANGSU GUOGUANG ELECTRONIC INFORMATION TECHNOLOGY
- KAL
- KEBA HANDOVER AUTOMATION GMBH
- NCR FSG
- NEXUS SOFTWARE
- OBERTHUR CASH PROTECTION
- OKI ELECTRIC INDUSTRY SHENZHEN
- SALZBURGER BANKEN SOFTWARE
- SECURE INNOVATION
- SIGMA SPA

It is possible that some elements of this CEN/CWA may be subject to patent rights. The CEN-CENELEC policy on patent rights is set out in CEN-CENELEC Guide 8 "Guidelines for Implementation of the Common IPR Policy on Patents (and other statutory intellectual property rights based on inventions)". CEN shall not be held responsible for identifying any or all such patent rights.
The Workshop participants have made every effort to ensure the reliability and accuracy of the technical and non-technical content of CWA 16926-12, but this does not guarantee, either explicitly or implicitly, its correctness. Users of CWA 16926-12 should be aware that neither the Workshop participants, nor CEN can be held liable for damages or losses of any kind whatsoever which may arise from its application. Users of CWA 16926-12 do so on their own responsibility and at their own risk.

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

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

Part 2: Service Classes Definition - Programmer's Reference

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

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

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

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

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

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

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

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

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

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

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

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

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

Part 16: Card Dispenser Device Class Interface - Programmer's Reference

Part 17: Barcode Reader Device Class Interface - Programmer's Reference

Part 18: Item Processing Module Device Class Interface - Programmer's Reference

Part 19: Biometrics Device Class Interface - Programmer's Reference

Parts 20 - 28: Reserved for future use.

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

Part 29: XFS MIB Architecture and SNMP Extensions - Programmer’s Reference

Part 30: XFS MIB Device Specific Definitions - Printer Device Class

Part 31: XFS MIB Device Specific Definitions - Identification Card Device Class

Part 32: XFS MIB Device Specific Definitions - Cash Dispenser Device Class

Part 33: XFS MIB Device Specific Definitions - PIN Keypad Device Class

Part 34: XFS MIB Device Specific Definitions - Check Reader/Scanner Device Class

Part 35: XFS MIB Device Specific Definitions - Depository Device Class

Part 36: XFS MIB Device Specific Definitions - Text Terminal Unit Device Class

Part 37: XFS MIB Device Specific Definitions - Sensors and Indicators Unit Device Class

Part 38: XFS MIB Device Specific Definitions - Camera Device Class

Part 39: XFS MIB Device Specific Definitions - Alarm Device Class

Part 40: XFS MIB Device Specific Definitions - Card Embossing Unit Class

Part 41: XFS MIB Device Specific Definitions - Cash-In Module Device Class

Part 42: Reserved for future use.

Part 43: XFS MIB Device Specific Definitions - Vendor Dependent Mode Device Class

Part 44: XFS MIB Application Management

Part 45: XFS MIB Device Specific Definitions - Card Dispenser Device Class

Part 46: XFS MIB Device Specific Definitions - Barcode Reader Device Class

Part 47: XFS MIB Device Specific Definitions - Item Processing Module Device Class

Part 48: XFS MIB Device Specific Definitions - Biometrics Device Class

Parts 49 - 60 are reserved for future use.

Part 61: Application Programming Interface (API) - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Service Provider Interface (SPI) - Programmer's Reference

Part 62: Printer and Scanning Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 63: Identification Card Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 65: PIN Keypad Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 67: Depository Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 71: Camera Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 72: Alarm Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 74: Cash-In Module Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 75: Card Dispenser Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 76: Barcode Reader Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 77: Item Processing Module Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (this CWA) - Programmer's Reference

Part 78: Biometric Device Class Interface - Migration from Version 3.40 (CWA 16296:2020) to Version 3.50 (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: https://www.cencenelec.eu/areas-of-work/cen-sectors/digital-society-cen/cwa-download-area/.

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

Revision History:

b3.10	November 29, 2007	Initial Release.
3.20	March 2, 2011	For a description of changes from version 3.10 to version 3.20 see the CAM 3.20 Migration document.
3.30	March 19, 2015	For a description of changes from version 3.20 to version 3.30 see the CAM 3.30 Migration document.
3.40	December 06, 2019	For a description of changes from version 3.30 to version 3.40 see the CAM 3.40 Migration document.
3.50	November 18, 2022	For a description of changes from version 3.40 to version 3.50 see the CAM 3.50 Migration document.

1. Introduction

1.1 Background to Release 3.50

The CEN/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 (European Committee for Standardization/Information Society Standardization System) Workshop environment. CEN Workshops aim to arrive at a European consensus on an issue that can be published as a CEN Workshop Agreement (CWA).

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

Release 3.50 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:

- Addition of E2E security
- PIN Password Entry

1.2 XFS Service-Specific Programming

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

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

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

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

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

The requested capability is defined for the class of Service Providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability ***is*** considered to be fundamental to the service. In this case, a WFS_ERR_UNSUPP_COMMAND error for Execute commands or WFS_ERR_UNSUPP_CATEGORY error for Info commands is returned to the calling application. An example would be a request from an application to a cash dispenser to retract items where the dispenser hardware does not have that capability; the Service Provider recognizes the command but, since the cash dispenser it is managing is unable to fulfil the request, 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 for Execute commands or WFS_ERR_INVALID_CATEGORY error for Info commands 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 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 **WFSGetInfo, WFSAsyncGetInfo**, **WFSExecute** and **WFSAsyncExecute** 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.~~40~~50

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;
     DWORD               dwMediaSize[WFS_CAM_CAMERAS_SIZE];
     } WFSCAMSTATUS, *LPWFSCAMSTATUS;
```

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

Value	Meaning
WFS_CAM_DEVONLINE	The device is online (i.e. powered on and operable).
WFS_CAM_DEVOFFLINE	The device is offline (e.g. the operator has taken the device offline by turning a switch).
WFS_CAM_DEVPOWEROFF	The device is powered off or physically not connected.
WFS_CAM_DEVNODEVICE	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.
WFS_CAM_DEVHWERROR	The device is inoperable due to a hardware error.
WFS_CAM_DEVUSERERROR	The device is inoperable because a person is preventing proper operation.
WFS_CAM_DEVBUSY	The device is busy and not able to process an execute command at this time.
WFS_CAM_DEVFRAUDATTEMPT	The device is present but is inoperable because it has detected a fraud attempt.
WFS_CAM_DEVPOTENTIALFRAUD	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.

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. For a device which stores pictures and videos on a hard disk drive or other general-purpose storage, the value of the *fwMedia* field should be WFS_CAM_MEDIANOTSUPP.

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:

Value	Meaning
WFS_CAM_MEDIAOK	The media is in a good state.
WFS_CAM_MEDIAHIGH	The media is almost full (threshold).
WFS_CAM_MEDIAFULL	The media is full.

WFS_CAM_MEDIANOTSUPP	The device does not support sensing the media level.
WFS_CAM_MEDIAUNKNOWN	Due to a hardware error or other condition, the state of the media cannot be determined.

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:

Value	Meaning
WFS_CAM_MEDIAOK	The media is in a good state.
WFS_CAM_MEDIAHIGH	The media is almost full (threshold).
WFS_CAM_MEDIAFULL	The media is full.
WFS_CAM_MEDIANOTSUPP	The device does not support sensing the media level.
WFS_CAM_MEDIAUNKNOWN	Due to a hardware error or other condition, the state of the media cannot be determined.

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:

Value	Meaning
WFS_CAM_MEDIAOK	The media is in a good state.
WFS_CAM_MEDIAHIGH	The media is almost full (threshold).
WFS_CAM_MEDIAFULL	The media is full.
WFS_CAM_MEDIANOTSUPP	The device does not support sensing the media level.
WFS_CAM_MEDIAUNKNOWN	Due to a hardware error or other condition, the state of the media cannot be determined.

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

Value	Meaning
WFS_CAM_CAMNOTSUPP	The camera is not supported.
WFS_CAM_CAMOK	The camera is in a good state.
WFS_CAM_CAMINOP	The camera is inoperative.
WFS_CAM_CAMUNKNOWN	Due to a hardware error or other condition, the state of the camera cannot be determined.
WFS_CAM_CAMRECORDING	The camera is recording a video.

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:

Value	Meaning
WFS_CAM_CAMNOTSUPP	The camera is not supported.
WFS_CAM_CAMOK	The camera is in a good state.
WFS_CAM_CAMINOP	The camera is inoperative.
WFS_CAM_CAMUNKNOWN	Due to a hardware error or other condition, the state of the camera cannot be determined.
WFS_CAM_CAMRECORDING	The camera is recording a video.

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:

Value	Meaning
WFS_CAM_CAMNOTSUPP	The camera is not supported.
WFS_CAM_CAMOK	The camera is in a good state.
WFS_CAM_CAMINOP	The camera is inoperative.

WFS_CAM_CAMUNKNOWN	Due to a hardware error or other condition, the state of the camera cannot be determined.
WFS_CAM_CAMRECORDING	The camera is recording a video.

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. For a device which stores pictures on a hard disk drive or other general-purpose storage, the value of the *usPictures* field should be zero.

Index	Meaning
WFS_CAM_ROOM	The camera that monitors the whole self-service area.
WFS_CAM_PERSON	The camera that monitors the person standing in front of the self-service machine.
WFS_CAM_EXITSLOT	The camera that monitors the exit slot(s) of the self-service machine.

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.

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

Value	Meaning
WFS_CAM_AFMNOTSUPP	No anti-fraud module is available.
WFS_CAM_AFMOK	Anti-fraud module is in a good state and no foreign device is detected.
WFS_CAM_AFMINOP	Anti-fraud module is inoperable.
WFS_CAM_AFMDEVICEDETECTED	Anti-fraud module detected the presence of a foreign device.
WFS_CAM_AFMUNKNOWN	The state of the anti-fraud module cannot be determined.

dwMediaSize […]
Specifies the total size(kilobytes) of pictures and videos stored on the recording media of the cameras.
A number of indexes are defined below. The maximum *dwMediaSize* index is WFS_CAM_CAMERAS_MAX. For a device which stores pictures and videos on a hard disk drive or other general-purpose storage, the value of the *dwMediaSize* field should be zero.

Value	Meaning
WFS_CAM_ROOM	The camera that monitors the whole self-service area.
WFS_CAM_PERSON	The camera that monitors the person standing in front of the self-service machine.
WFS_CAM_EXITSLOT	The camera that monitors the exit slot(s) of the self-service machine.

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;

```
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;
     LPDWORD             lpdwSynchronizableCommands;
     LPWFSCAMDETAIL      *lppCameraDetail;
     DWORD               fwVideoFormatSupport;
     DWORD               dwMaxMediaSize;
     } 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:

Value	Meaning
WFS_CAM_TYPE_CAM	Camera system.

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:

Value	Meaning
WFS_CAM_NOT_AVAILABLE	This camera is not available.
WFS_CAM_AVAILABLE	This camera is available.

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:

Value	Meaning
WFS_CAM_NOT_AVAILABLE	This camera is not available.
WFS_CAM_AVAILABLE	This camera is available.

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

Value	Meaning
WFS_CAM_NOT_AVAILABLE	This camera is not available.
WFS_CAM_AVAILABLE	This camera is available.

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 or video. Specified as a combination of the following flags:

Value	Meaning
WFS_CAM_NOTADD	No data can be added to the picture or video.
WFS_CAM_AUTOADD	Data is added automatically to the picture or video.
WFS_CAM_MANADD	Data can be added manually to the picture or video using the field *lpszCamData* in the WFS_CMD_CAM_TAKE_PICTURE, WFS_CMD_CAM_TAKE_PICTURE_EX, WFS_CMD_CAM_TAKE_PICTURE_350, WFS_CMD_CAM_TAKE_VIDEO command.

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

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

Value	Meaning
WFS_CAM_ASCII	ASCII is supported for execute command data values.
WFS_CAM_UNICODE	UNICODE is supported for execute command data values.

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.

lpdwSynchronizableCommands
Pointer to a zero-terminated list of DWORDs which contains the execute command IDs that can be synchronized. If no execute command can be synchronized then this parameter will be NULL.

lppCameraDetail
Pointer to a null-terminated array of pointers to WFSCAMDETAIL structures, describing the detail of the cameras. The maximum number of the structures is WFS_CAM_CAMERAS_MAX. If a camera dose not report this information, then the camera does not support the video function and the WFS_CMD_CAM_TAKE_VIDEO command cannot be used:

```
typedef struct _wfs_cam_detail
      {
      WORD                        wCamera;
      WORD                        fwMediaType;
      LPWFSCAMPIXELSIZE           lpMaxPixelSize;
      USHORT                      usMaxFps;
      DWORD                       dwKbps;
      } WFSCAMDETAIL, *LPWFSCAMDETAIL;
```

wCamera
Specifies the camera as one of the following flags:

Value	Meaning
WFS_CAM_ROOM	Monitors the whole self-service area.
WFS_CAM_PERSON	Monitors the person standing in front of the self-service machine.
WFS_CAM_EXITSLOT	Monitors the exit slot(s) of the self-service machine.

fwMediaType
Specifies the supported media type of the camera as a combination of the following flags:

Value	Meaning
WFS_CAM_MEDIAPICTURE	The camera can take pictures by using WFS_CMD_CAM_TAKE_PICTURE, WFS_CMD_CAM_TAKE_PICTURE_EX or WFS_CMD_CAM_TAKE_PICTURE_350.
WFS_CAM_MEDIAVIDEO	The camera can take video by using WFS_CMD_CAM_TAKE_VIDEO.

lpMaxPixelSize
Pointer to a WFSCAMPIXELSIZE structure, describing the maximum pixel size of the camera:

```
typedef struct _wfs_cam_pixelsize
      {
      ULONG                         ulSizeX;
      ULONG                         ulSizeY;
      } WFSCAMPIXELSIZE, *LPWFSCAMPIXELSIZE;
```

ulSizeX
Specifies the pixel size of the width.

ulSizeY
Specifies the pixel size of the height.

usMaxFps
Specifies the maximum frame rate of the camera. If WFS_CAM_MEDIAVIDEO is not specified in *fwMediaType*, this parameter will be 0.

dwKbps
Specifies the bitrate of the camera by kilobits per second. If WFS_CAM_MEDIAVIDEO is not specified in *fwMediaType*, or the Service Provider cannot get the bitrate from the camera, this parameter will be 0.

fwVideoFormatSupport
Specifies the supported video format as a combination of the following flags. Any file extensions of a supported format can be specified in the WFS_CMD_CAM_TAKE_VIDEO command. If WFS_CAM_MEDIAVIDEO is not specified in all camera's *fwMediaType*, this parameter will be 0.

Value	Meaning
WFS_CAM_VIDEOFORMAT_AVI	The AVI video format is supported.
WFS_CAM_VIDEOFORMAT_MOV	The MOV video format is supported.
WFS_CAM_VIDEOFORMAT_WMV	The WMV video format is supported.
WFS_CAM_VIDEOFORMAT_FLV	The FLV video format is supported.
WFS_CAM_VIDEOFORMAT_MPEG	The MPEG video format is supported.
WFS_CAM_VIDEOFORMAT_MPEG2	The MPEG2 video format is supported.
WFS_CAM_VIDEOFORMAT_MPEG4	The MPEG4 video format is supported.
WFS_CAM_VIDEOFORMAT_MKV	The MKV video format is supported.
WFS_CAM_VIDEOFORMAT_ASF	The ASF video format is supported.
WFS_CAM_VIDEOFORMAT_VOB	The VOB video format is supported.

dwMaxMediaSize
Specifies the maximum size(kilobytes) of pictures and videos that can be stored on the recording media.

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 *lpszCamData* or *lpszUNICODECamData* parameter.

Input Param LPWFSCAMTAKEPICT lpTakePict;

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

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

Value	Meaning
WFS_CAM_ROOM	Monitors the whole self-service area.
WFS_CAM_PERSON	Monitors the person standing in front of the self-service machine.
WFS_CAM_EXITSLOT	Monitors the exit slot(s) of the self-service machine.

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.

The function strftime's directives that begin with a percent (%) character can be contained in this parameter to represent the formats of variable dates and times.

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

The function strftime's directives that begin with a percent (%) character can be contained in this parameter to represent the formats of variable dates and times.

Output Param None.

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

Value	Meaning
WFS_ERR_CAM_CAMNOTSUPP	The specified camera is not supported.
WFS_ERR_CAM_MEDIAFULL	The recording media is full.
WFS_ERR_CAM_CAMINOP	The specified camera is inoperable.
WFS_ERR_CAM_CHARSETNOTSUPP	Character set(s) supported by Service Provider is inconsistent with use of *lpszCamData* or *lpszUNICODECamData* fields.
WFS_ERR_CAM_FILEIOERROR	Directory does not exist or File IO error while storing the image to the hard disk.

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

Value	Meaning
WFS_USRE_CAM_MEDIATHRESHOLD	The state of the recording media reached a threshold.
WFS_EXEE_CAM_INVALIDDATA	The text string given is too long or in some other way invalid.

Comments None.

5.2 WFS_CMD_CAM_RESET

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

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 LPWFSCAMTAKEPICTEX lpTakePictEx;

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

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

Value	Meaning
WFS_CAM_ROOM	Monitors the whole self-service area.
WFS_CAM_PERSON	Monitors the person standing in front of the self-service machine.
WFS_CAM_EXITSLOT	Monitors the exit slot(s) of the self-service machine.

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.

The function strftime's directives that begin with a percent (%) character can be contained in this parameter to represent the formats of variable dates and times.

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

The function strftime's directives that begin with a percent (%) character can be contained in this parameter to represent the formats of variable dates and times.

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:

Value	Meaning
WFS_ERR_CAM_CAMNOTSUPP	The specified camera is not supported.
WFS_ERR_CAM_MEDIAFULL	The recording media is full.
WFS_ERR_CAM_CAMINOP	The specified camera is inoperable.
WFS_ERR_CAM_CHARSETNOTSUPP	Character set(s) supported by Service Provider is inconsistent with use of *lpszCamData* or *lpszUNICODECamData* fields.
WFS_ERR_CAM_FILEIOERROR	Directory does not exist or File IO error while storing the image to the hard disk.

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

command:

Value	Meaning
WFS_USRE_CAM_MEDIATHRESHOLD	The state of the recording media reached a threshold.
WFS_EXEE_CAM_INVALIDDATA	The text string given is too long or in some other way invalid.

Comments None.

5.4 WFS_CMD_CAM_SYNCHRONIZE_COMMAND

Description This command is used to reduce response time of a command (e.g. for synchronization with display) as well as to synchronize actions of the different device classes. This command is intended to be used only on hardware which is capable of synchronizing functionality within a single device class or with other device classes.

The list of execute commands which this command supports for synchronization is retrieved in the *lpdwSynchronizableCommands* parameter of the WFS_INF_CAM_CAPABILITIES.

This command is optional, i.e. any other command can be called without having to call it in advance. Any preparation that occurs by calling this command will not affect any other subsequent command. However, any subsequent execute command other than the one that was specified in the *dwCommand* input parameter will execute normally and may invalidate the pending synchronization. In this case the application should call the WFS_CMD_CAM_SYNCHRONIZE_COMMAND again in order to start a synchronization.

Input Param LPWFSCAMSYNCHRONIZECOMMAND lpSynchronizeCommand;

```
typedef struct _wfs_cam_synchronize_command
     {
     DWORD                    dwCommand;
     LPVOID                   lpCmdData;
     } WFSCAMSYNCHRONIZECOMMAND, *LPWFSCAMSYNCHRONIZECOMMAND;
```

dwCommand
The command ID of the command to be synchronized and executed next.

lpCmdData
Pointer to data or a data structure that represents the parameter that is normally associated with the command that is specified in *dwCommand*. For example, if *dwCommand* is WFS_CMD_CAM_TAKE_PICTURE then *lpCmdData* will point to a WFSCAMTAKEPICT structure. This parameter can be NULL if no command input parameter is needed or if this detail is not needed to synchronize for the command.

It will be device-dependent whether the synchronization is effective or not in the case where the application synchronizes for a command with this command specifying a parameter but subsequently executes the synchronized command with a different parameter. This case should not result in an error; however, the preparation effect could be different from what the application expects. The application should, therefore, make sure to use the same parameter between *lpCmdData* of this command and the subsequent corresponding execute command.

Output Param None.

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

Value	Meaning
WFS_ERR_CAM_COMMANDUNSUPP	The command specified in the *dwCommand* field is not supported by the Service Provider.
WFS_ERR_CAM_SYNCHRONIZEUNSUPP	The preparation for the command specified in the *dwCommand* with the parameter specified in the *lpCmdData* is not supported by the Service Provider.

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

Comments None.

5.5 WFS_CMD_CAM_TAKE_PICTURE_350

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. *lppCaptions* can not only specify the text string to be displayed, but also specify the start position of the text string. So different text string can be displayed on different positions by using *lppCaptions*. The pixel size of the output picture file can be scaled with the same ratio of the maximum pixel size reported in the *lppCameraDetail* parameter of WFS_INF_CAM_CAPABILITIES command by using the *lpPixelSize* parameter.

Input Param LPWFSCAMTAKEPICTEX lpTakePict350;

```
typedef struct _wfs_cam_take_picture_350
      {
      WORD                    wCamera;
      LPSTR                   lpszPictureFile;
      LPWFSCAMPIXELSIZE       lpPixelSize;
      LPWFSCAMCAP             *lppCaptions;
      } WFSCAMTAKEPICT350, *LPWFSCAMTAKEPICT350;
```

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

Value	Meaning
WFS_CAM_ROOM	Monitors the whole self-service area.
WFS_CAM_PERSON	Monitors the person standing in front of the self-service machine.
WFS_CAM_EXITSLOT	Monitors the exit slot(s) of the self-service machine.

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.

lpPixelSize
Pointer to a WFSCAMPIXELSIZE structure, specifying the pixel size of the output picture's width and height. The width/height ratio of *lpPixelSize* should be the same as the ratio of maximum pixel reported in the *lppCameraDetail* parameter of WFS_INF_CAM_CAPABILITIES command. If the ratio is different from the maximum pixel, the larger of width and height will be used as one side of the output picture, and the other side will be scaled by the same ratio of maximum pixel. This size cannot be larger than the maximum pixel size. Even if only one side's pixel size is larger than the maximum pixel size. *lpPixelSize* will be treated as NULL. If set NULL, the maximum pixel size will be used.

lppCaptions
Pointer to an array of null-terminated pointers to WFSCAMCAP structures, specifying the text string, start position of the caption displayed on the picture:

```
typedef struct _wfs_cam_cap
      {
      LPSTR                     lpszCamData;
      LPWSTR                    lpszUNICODECamData;
      LPWFSCAMCAPPOS            lpCamDataPosition;
      } WFSCAMCAP, *LPWFSCAMCAP;
```

lpszCamData
Specifies the text string to be displayed on the picture. 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 function strftime's directives that begin with a percent (%) character can be contained in this parameter to represent the formats of variable dates and times.

lpszUNICODECamData
Specifies the UNICODE text string to be displayed on the picture. 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 *lpszUNICODECamData* field should only be used if the Service Provider supports UNICODE. The lpszCamData and lpszUNICODECamData fields are mutually exclusive.

The function strftime's directives that begin with a percent (%) character can be contained in this parameter to represent the formats of variable dates and times.

lpCamDataPosition
Pointer to a WFSCAMCAPPOS structure, specifying the start position of the text string specified by *lpszCamData* or *lpszUNICODECamData* in WFSCAMCAP structure on the picture. The bottom left corner of the text string box corresponds to the start position. If the specified position is out of the camera's pixel size, the text string will be not displayed on the picture. In this case, an execute event WFS_EXEE_CAM_INVALIDDATA is generated. Nevertheless the picture is taken. The top left corner of the picture is the point 0,0.

```
typedef struct _wfs_cam_cappos
    {
    ULONG                   ulPositionX;
    ULONG                   ulPositionY;
    } WFSCAMCAPPOS, *LPWFSCAMCAPPOS;
```

ulPositionX
Specifies the width coordinates(pixel) of the start position.

ulPositionY
Specifies the height coordinates(pixel) of the start position.

Output Param None.

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

Value	Meaning
WFS_ERR_CAM_CAMNOTSUPP	The specified camera is not supported.
WFS_ERR_CAM_MEDIAFULL	The recording media is full.
WFS_ERR_CAM_CAMINOP	The specified camera is inoperable.
WFS_ERR_CAM_CHARSETNOTSUPP	Character set(s) supported by Service Provider is inconsistent with use of *lpszCamData* or *lpszUNICODECamData* fields.
WFS_ERR_CAM_FILEIOERROR	Directory does not exist or File IO error while storing the image to the hard disk.

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

Value	Meaning
WFS_USRE_CAM_MEDIATHRESHOLD	The state of the recording media reached a threshold.
WFS_EXEE_CAM_INVALIDDATA	The text string given is too long or in some other way invalid.

Comments None.

5.6 WFS_CMD_CAM_TAKE_VIDEO

Description This command is used to take a video. It is possible to select which camera or which camera position should be used to take a video. This command can be used to save the recorded video as a file. If *dwTimeOut* of **WFPExecute** is zero, the Service Provider does not terminate the command unless the application issues a **WFSCancelAsyncRequest** command. The parameter *lpPixelSize* and *usFps* can be used to specify the pixel size and the frame rate of the video. Text string and start position of captions displayed on the video can be specified using the parameter *lppCaptions*.

Input Param LPWFSCAMTAKEVIDEO lpTakeVideo;

```
typedef struct _wfs_cam_take_video
      {
      WORD                    wCamera;
      LPWFSCAMPIXELSIZE       lpPixelSize;
      USHORT                  usFps;
      LPSTR                   lpszVideoFile;
      LPWFSCAMCAP             *lppCaptions;
      } WFSCAMTAKEVIDEO, *LPWFSCAMTAKEVIDEO;
```

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

Value	Meaning
WFS_CAM_ROOM	Monitors the whole self-service area.
WFS_CAM_PERSON	Monitors the person standing in front of the self-service machine.
WFS_CAM_EXITSLOT	Monitors the exit slot(s) of the self-service machine.

lpPixelSize
Pointer to a WFSCAMPIXELSIZE structure, specifying the pixel size of the output video's width and height. The width/height ratio of *lpPixelSize* should be the same as the ratio of maximum pixel reported in the *lppCameraDetail* parameter of WFS_INF_CAM_CAPABILITIES command. If the ratio is different from the maximum pixel, the larger of width and height will be used as one side of the output video, and the other side will be scaled by the same ratio of maximum pixel. This size cannot be larger than the maximum pixel size. Even if only one side's pixel size is larger than the maximum pixel size. *lpPixelSize* will be treated as NULL. If set NULL, the maximum pixel size will be used.

usFps
Specifies the frame rate to be used when taking video. This value cannot be larger than the maximum frame rate that is reported in the parameter *usMaxFps* of the pointer *lppCameraDetail* in WFS_INF_CAM_CAPABILITIES command. If set 0, the default frame rate will be used.

lpszVideoFile
Specifies the full path and file name of the video to be taken by a camera device. This value is terminated with a single null character and cannot contain UNICODE characters. The file name includes the video format specific file extension. The supported video formats are reported in the parameter *fwVideoFormatSupport* of WFS_INF_CAM_CAPABILITIES command. The Service Provider is responsible for converting the video into the required format. If the extension is not specified or an unsupported extension is specified, the Service Provider will use the default format.

lppCaptions
Pointer to an array of null-terminated pointers to WFSCAMCAP structures, specifying the text string, start position of the caption displayed on the video. For the WFSCAMCAP structure specification see the definition of the command WFS_CMD_CAM_TAKE_PICTURE_350.

Output Param None.

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

Value	Meaning
WFS_ERR_CAM_CAMNOTSUPP	The specified camera is not supported.
WFS_ERR_CAM_MEDIAFULL	The recording media is full.

WFS_ERR_CAM_CAMINOP	The specified camera is inoperable.
WFS_ERR_CAM_CHARSETNOTSUPP	Character set(s) supported by Service Provider is inconsistent with use of *lpszCamData* or *lpszUNICODECamData* fields.
WFS_ERR_CAM_FILEIOERROR	Directory does not exist or File IO error while storing the video to the hard disk.
WFS_ERR_CAM_CAMRECORDING	The specified camera is recording a video.
WFS_ERR_CAM_PIXELSIZENOTSUPP	The specified pixel size is not supported.
WFS_ERR_CAM_FPSNOTSUPP	The specified frame rate is not supported.

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

Value	Meaning
WFS_USRE_CAM_MEDIATHRESHOLD	The state of the recording media reached a threshold.
WFS_EXEE_CAM_INVALIDDATA	The text string given is too long or in some other way invalid.
WFS_USRE_CAM_VIDEOSTART	Notify the application that the video has started.
WFS_USRE_CAM_VIDEOEND	Notify the application that the video has finished.
WFS_USRE_CAM_VIDEOERROR	Notify the application that the camera occurred an error when taking a video.

Comments None.

6. Events

6.1 WFS_USRE_CAM_MEDIATHRESHOLD

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~~A pointer to one of the following flags:

Value	Meaning
WFS_CAM_MEDIAOK	The recording media is a good state.
WFS_CAM_MEDIAHIGH	The recording media is almost full.
WFS_CAM_MEDIAFULL	The recording media is full.

Comments None.

6.2 WFS_EXEE_CAM_INVALIDDATA

Description This execute event is used to specify that the text string given was too long or in some other way invalid.

Event Param None.

Comments None.

6.3 WFS_USRE_CAM_VIDEOSTART

Description This user event is used to notify the application that the video has started.

Event Param LPWORD lpwCamera;

lpwCamera
A pointer to a flag specifying which camera has started taking video:

Value	Meaning
WFS_CAM_ROOM	Monitors the whole self-service area.
WFS_CAM_PERSON	Monitors the person standing in front of the self-service machine.
WFS_CAM_EXITSLOT	Monitors the exit slot(s) of the self-service machine.

Comments None.

6.4 WFS_USRE_CAM_VIDEOEND

Description This user event is used to notify the application that the video has finished.

Event Param LPWORD lpwCamera;

lpwCamera
A pointer to a flag specifying which camera has started taking video:

Value	Meaning
WFS_CAM_ROOM	Monitors the whole self-service area.
WFS_CAM_PERSON	Monitors the person standing in front of the self-service machine.
WFS_CAM_EXITSLOT	Monitors the exit slot(s) of the self-service machine.

Comments None.

6.5 WFS_USRE_CAM_VIDEOERROR

Description This user event is used to notify the application that an error occurred while taking a video.

Event Param LPWFSCAMVIDEOERROR lpVideoError;

```
typedef struct _wfs_cam_video_error
      {
      WORD                        wCamera;
      WORD                        wFailure;
      } WFSCAMVIDEOERROR, *LPWFSCAMVIDEOERROR;
```

wCamera
Specified which camera finish taking video as one of the following:

Value	Meaning
WFS_CAM_ROOM	Monitors the whole self-service area.
WFS_CAM_PERSON	Monitors the person standing in front of the self-service machine.
WFS_CAM_EXITSLOT	Monitors the exit slot(s) of the self-service machine.

wFailure
Specifies the kind of failure that occurred when taking a video. Values are:

Value	Meaning
WFS_CAM_FAIL_MEDIAFULL	The recording media is full.
WFS_CAM_FAIL_FILEIOERROR	Directory does not exist or File IO error while storing the video to the hard disk.

Comments None.

7. C - Header file

```
/******************************************************************************
*                                                                             *
* xfscam.h      XFS - Camera (CAM) definitions                                *
*                                                                             *
*             Version 3.40  (December 6 2019) 50  (November 18 2022)          *
*                                                                             *
*                                                                             *
******************************************************************************/

#ifndef __INC_XFSCAM__H
#define __INC_XFSCAM__H

#ifdef __cplusplus
extern "C" {
#endif

#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             (0x28030x3203) /* Version 3.4050 */
#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)
#define     WFS_CMD_CAM_SYNCHRONIZE_COMMAND     (CAM_SERVICE_OFFSET + 4)
#define     WFS_CMD_CAM_TAKE_PICTURE_350        (CAM_SERVICE_OFFSET + 5)
#define     WFS_CMD_CAM_TAKE_VIDEO              (CAM_SERVICE_OFFSET + 6)

/* CAM Messages */

#define     WFS_USRE_CAM_MEDIATHRESHOLD         (CAM_SERVICE_OFFSET + 1)
#define     WFS_EXEE_CAM_INVALIDDATA            (CAM_SERVICE_OFFSET + 2)
#define     WFS_USRE_CAM_VIDEOSTART             (CAM_SERVICE_OFFSET + 3)
#define     WFS_USRE_CAM_VIDEOEND               (CAM_SERVICE_OFFSET + 4)
#define     WFS_USRE_CAM_VIDEOERROR             (CAM_SERVICE_OFFSET + 5)

/* values of WFSCAMSTATUS.fwDevice */

#define     WFS_CAM_DEVONLINE                   WFS_STAT_DEVONLINE
#define     WFS_CAM_DEVOFFLINE                  WFS_STAT_DEVOFFLINE
#define     WFS_CAM_DEVPOWEROFF                 WFS_STAT_DEVPOWEROFF
#define     WFS_CAM_DEVNODEVICE                 WFS_STAT_DEVNODEVICE
#define     WFS_CAM_DEVHWERROR                  WFS_STAT_DEVHWERROR
#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 [...]
              WFSCAMSTATUS.usPictures[...]
              WFSCAMCAPS.fwCameras [...]
              WFSCAMTAKEPICT.wCamera             */

#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)
#define     WFS_CAM_CAMRECORDING                (4)

/* 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)

/* values of WFSCAMCAPS.lppCameraDetail.fwMediaType */

#define     WFS_CAM_MEDIAPICTURE                (0x0001)
#define     WFS_CAM_MEDIAVIDEO                  (0x0002)

/* flags of WFSCAMCAPS.fwVideoFormatSupport */

#define     WFS_CAM_VIDEOFORMAT_AVI             (0x0001)
#define     WFS_CAM_VIDEOFORMAT_MOV             (0x0002)
#define     WFS_CAM_VIDEOFORMAT_WMV             (0x0004)
#define     WFS_CAM_VIDEOFORMAT_FLV             (0x0008)
#define     WFS_CAM_VIDEOFORMAT_MPEG            (0x0010)
#define     WFS_CAM_VIDEOFORMAT_MPEG2           (0x0020)
#define     WFS_CAM_VIDEOFORMAT_MPEG4           (0x0040)
#define     WFS_CAM_VIDEOFORMAT_MKV             (0x0080)
```

```
#define     WFS_CAM_VIDEOFORMAT_ASF             (0x0100)
#define     WFS_CAM_VIDEOFORMAT_VOB             (0x0200)

/* values of WFSCAMVIDEOERROR.wFailure */

#define     WFS_CAM_FAIL_MEDIAFULL              (0)
#define     WFS_CAM_FAIL_FILEIOERROR            (1)

/* XFS CAM Errors */

#define WFS_ERR_CAM_CAMNOTSUPP                  (-(CAM_SERVICE_OFFSET + 0))
#define WFS_ERR_CAM_MEDIAFULL                   (-(CAM_SERVICE_OFFSET + 1))
#define WFS_ERR_CAM_CAMINOP                     (-(CAM_SERVICE_OFFSET + 2))
#define WFS_ERR_CAM_CHARSETNOTSUPP              (-(CAM_SERVICE_OFFSET + 3))
#define WFS_ERR_CAM_FILEIOERROR                 (-(CAM_SERVICE_OFFSET + 4))
#define WFS_ERR_CAM_COMMANDUNSUPP               (-(CAM_SERVICE_OFFSET + 5))
#define WFS_ERR_CAM_SYNCHRONIZEUNSUPP           (-(CAM_SERVICE_OFFSET + 6))
#define WFS_ERR_CAM_CAMRECORDING                (-(CAM_SERVICE_OFFSET + 7))
#define WFS_ERR_CAM_PIXELSIZENOTSUPP            (-(CAM_SERVICE_OFFSET + 8))
#define WFS_ERR_CAM_FPSNOTSUPP                  (-(CAM_SERVICE_OFFSET + 9))

/*=================================================================*/
/* CAM Info Command Structures */
/*=================================================================*/

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;
    DWORD               ____dwMediaSize[WFS_CAM_CAMERAS_SIZE];
} WFSCAMSTATUS, *LPWFSCAMSTATUS;

typedef struct _wfs_cam_pixelsize
{
    ULONG               ____ulSizeX;
    ULONG               ____ulSizeY;
} WFSCAMPIXELSIZE, *LPWFSCAMPIXELSIZE;

typedef struct _wfs_cam_detail
{
    WORD                ____wCamera;
    WORD                ____fwMediaType;
    LPWFSCAMPIXELSIZE   ____lpMaxPixelSize;
    USHORT              ____usMaxFps;
    DWORD               ____dwKbps;
} WFSCAMDETAIL, *LPWFSCAMDETAIL;

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;
    LPDWORD             ____lpdwSynchronizableCommands;
    LPWFSCAMDETAIL      ____*lppCameraDetail;
    DWORD               ____fwVideoFormatSupport;
    DWORD               ____dwMaxMediaSize;
} WFSCAMCAPS, *LPWFSCAMCAPS;

/*=================================================================*/
```

```
/* CAM Execute Command Structures */
/*=====================================================================*/

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;

typedef struct _wfs_cam_synchronize_command
{
    DWORD             dwCommand;
    LPVOID            lpCmdData;
} WFSCAMSYNCHRONIZECOMMAND, *LPWFSCAMSYNCHRONIZECOMMAND;

typedef struct _wfs_cam_cappos
{
    ULONG             ulPositionX;
    ULONG             ulPositionY;
} WFSCAMCAPPOS, *LPWFSCAMCAPPOS;

typedef struct _wfs_cam_cap
{
    LPSTR             lpszCamData;
    LPSTR             lpszUNICODECamData;
    LPWFSCAMCAPPOS    lpCamDataPosition;
} WFSCAMCAP, *LPWFSCAMCAP;

typedef struct _wfs_cam_take_picture_350
{
    WORD              wCamera;
    LPSTR             lpszPictureFile;
    LPWFSCAMPIXELSIZE lpPixelSize;
    LPWFSCAMCAP       *lppCaptions;
} WFSCAMTAKEPICT350, *LPWFSCAMTAKEPICT350;

typedef struct _wfs_cam_take_video
{
    WORD              wCamera;
    LPWFSCAMPIXELSIZE lpPixelSize;
    USHORT            usFps;
    LPSTR             lpszVideoFile;
    LPWFSCAMCAP       *lppCaptions;
} WFSCAMTAKEVIDEO, *LPWFSCAMTAKEVIDEO;

/*=====================================================================*/
/* CAM Event Structures */
/*=====================================================================*/

typedef struct _wfs_cam_video_error
{
    WORD              wCamera;
    WORD              wFailure;
} WFSCAMVIDEOERROR, *LPWFSCAMVIDEOERROR;

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

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

```
#endif  /* __INC_XFSCAM__H */
```