CEN

WORKSHOP

AGREEMENT

CWA 15748-12

July 2008

ICS 35.240.50

English version

Extensions for Financial Services (XFS) interface specification - Release 3.10 - Part 12: Camera Device Class Interface - Programmer's Reference

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

Management Centre: rue de Stassart, 36 B-1050 Brussels

Ref. No.:CWA 15748-12:2008 D/E/F

Table of Contents

Foreword

This CWA is revision 3.10 of the XFS interface specification.

The CEN/ISSS XFS Workshop gathers suppliers as well as banks and other financial service companies. A list of companies participating in this Workshop and in support of this CWA is available from the CEN/ISSS Secretariat.

This CWA was formally approved by the XFS Workshop meeting on 2007-11-29. 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 3.10.

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

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 - 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

Parts 48 - 60 are reserved for future use.

Part 61: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 62: Printer Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 63: Identification Card Device Class Interface - Migration from Version 3.02 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 65: PIN Keypad Device Class Interface - Migration from Version 3.03 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 67: Depository Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.01 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 71: Camera Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 72: Alarm Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.0 (CWA 14050) to Version 3.10 (this CWA) - Programmer's Reference

Part 74: Cash-In Module Device Class Interface - Migration from Version 3.02 (CWA 14050) to Version 3.10 (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/isss/Workshop/XFS.

Revision History:

1.0	May 24, 1993	Initial release of API and SPI specification.
1.11	February 3, 1995	Separation of specification into separate documents for API/SPI and service class definitions.
2.0	November 11, 1996	Update release encompassing the self-service environment.
3.0	October 18, 2000	Update release encompassing: • Added WFS_CMD_CAM_RESET • UNICODE support For a detailed description see CWA 14050-24 CAM migration document from version 2.0 to version 3.0, revision 1.0, October 18th 2000.
3.10	November 29, 2007	For a description of changes see CWA 15748-71:2007 CAM Migration from Version 3.0 (see CWA 14050) to Version 3.10.

1. Introduction

1.1 Background to Release 3.10

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 **WFSExecute** function, the commands to read data from various services are as similar as possible to each other in their syntax and data structures.

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

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

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

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

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

This design allows implementation of applications that can be used with a range of services that provide differing subsets of the functionalities that are defined for their service class. Applications may use the **WFSGetInfo** and **WFSAsyncGetInfo** commands to inquire about the capabilities of the service they are about to use, and modify their 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 **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 3.10

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;
     } 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 or pulling out the device).
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 has detected a fraud attempt.

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:

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.

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.

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.

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.

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.

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;
     } 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. Specified as a combination of the following flags:

Value	Meaning
WFS_CAM_NOTADD	No data can be added to the picture.
WFS_CAM_AUTOADD	Data is added automatically to the picture.

WFS_CAM_MANADD	Data can be added manually to the picture using the filed *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:

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.

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. Furthermore data can be sent to be displayed on the photo.

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.

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.

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.

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 to 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.

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 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.

7. C - Header file

```
/******************************************************************************
*                                                                             *
* xfscam.h      XFS - Camera (CAM) definitions                                *
*                                                                             *
*               Version 3.10 (29/11/2007)                                     *
*                                                                             *
******************************************************************************/

#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             (0x0A03) /* Version 3.10 */
#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)

/* 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_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

/* 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.fwPictures[...]
              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)

/* 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, WFSCAMTAKEPICT.fwCharSupport  */

#define     WFS_CAM_ASCII                       (0x0001)
#define     WFS_CAM_UNICODE                     (0x0002)

/* 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))

/*=================================================================*/
/* 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;
} WFSCAMSTATUS, *LPWFSCAMSTATUS;

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;
} WFSCAMCAPS, *LPWFSCAMCAPS;
```

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

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

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

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

#endif  /* __INC_XFSCAM__H */
```