CEN

WORKSHOP

AGREEMENT

CWA 16926-67

January 2023

English version

Extensions for Financial Services (XFS) interface specification Release 3.50 - Part 67: Depository Device Class Interface - Programmer's Reference - Migration from Version 3.40 (CWA 16926:2020) to Version 3.50 (this CWA)

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

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

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

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

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

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

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

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 – Part 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 16926-8, but this does not guarantee, either explicitly or implicitly, its correctness. Users of CWA 16926-8 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 16926-8 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:

3.00	October 18, 2000	Initial Release.
3.10	November 29, 2007	For a description of changes from version 3.00 to version 3.10 see the DEP 3.10 Migration document.
3.20	March 2, 2011	For a description of changes from version 3.10 to version 3.20 see the DEP 3.20 Migration document.
3.30	March 19, 2015	For a description of changes from version 3.20 to version 3.30 see the DEP 3.30 Migration document.
3.40	December 06, 2019	For a description of changes from version 3.30 to version 3.40 see the DEP 3.40 Migration document.
3.50	November 18, 2022	For a description of changes from version 3.40 to version 3.50 see the DEP 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. Depository Unit

This specification describes the functionality of the services provided by the Depository (DEP) services under XFS, by defining the service-specific commands that can be issued, using the **WFSGetInfo, WFSAsyncGetInfo, WFSExecute** and **WFSAsyncExecute** functions.

A Depository is used for the acceptance and deposit of media into the device or terminal. There are two main types of depository: an envelope depository for the deposit of media in envelopes and a night safe depository for the deposit of bags containing bulk media.

An envelope depository accepts media, prints on the media and deposits the media into a holding container or bin. Some envelope depositories offer the capability to dispense an envelope to the customer at the start of a transaction. The customer takes this envelope, fills in the deposit media, possibly inscribes it and puts it into the deposit slot. The envelope is then accepted, printed and transported into a deposit container.

The envelope dispense mechanism may be part of the envelope depository device mechanism with the same entry/exit slot or it may be a separate mechanism with separate entry/exit slot.

Envelopes dispensed and not taken by the customer can be retracted back into the device. When the dispenser is a separate mechanism the envelope is retracted back into the dispenser container. When the dispenser is a common mechanism the envelope is retracted into the depository container.

A night safe depository normally only logs the deposit of a bag and does not print on the media.

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_DEP_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   LPWFSDEPSTATUS lpStatus;

```
typedef struct _wfs_dep_status
    {
    WORD              fwDevice;
    WORD              fwDepContainer;
    WORD              fwDepTransport;
    WORD              fwEnvSupply;
    WORD              fwEnvDispenser;
    WORD              fwPrinter;
    WORD              fwToner;
    WORD              fwShutter;
    WORD              wNumOfDeposits;
    LPSTR             lpszExtra;
    DWORD             dwGuidLights[WFS_DEP_GUIDLIGHTS_SIZE];
    WORD              fwDepositLocation;
    WORD              wDevicePosition;
    USHORT            usPowerSaveRecoveryTime;
    WORD              wAntiFraudModule;
    } WFSDEPSTATUS, *LPWFSDEPSTATUS;
```

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

Value	Meaning
WFS_DEP_DEVONLINE	The device is online (i.e. powered on and operable).
WFS_DEP_DEVOFFLINE	The device is off-line (e.g. the operator has taken the device offline by turning a switch).
WFS_DEP_DEVPOWEROFF	The device is powered off or physically not connected.
WFS_DEP_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_DEP_DEVHWERROR	The device is inoperable due to a hardware error. The device is present but a hardware fault prevents it from being used.
WFS_DEP_DEVUSERERROR	The device is present but a person is preventing proper operation. The application should suspend the device operation or remove the device from service until the Service Provider generates a device state change event indicating the condition of the device has changed, i.e. the error is removed or a permanent error condition has occurred.
WFS_DEP_DEVBUSY	The device is busy and not able to process an Execute command at this time.
WFS_DEP_DEVFRAUDATTEMPT	The device is present but is inoperable because it has detected a fraud attempt.
WFS_DEP_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.

fwDepContainer

Specifies the state of the deposit container that contains the deposited envelopes or bags as one of the following flags:

Value	Meaning
WFS_DEP_DEPOK	The deposit container is in a good state.
WFS_DEP_DEPHIGH	The deposit container is almost full (threshold).
WFS_DEP_DEPFULL	The deposit container is full.
WFS_DEP_DEPINOP	The deposit container is inoperable.
WFS_DEP_DEPMISSING	The deposit container is missing.
WFS_DEP_DEPUNKNOWN	Due to a hardware error or other condition, the state of the deposit container cannot be determined.
WFS_DEP_DEPNOTSUPP	The physical device is not able to determine the status of the deposit container.

fwDepTransport

Specifies the state of the deposit transport mechanism that transports the envelope into the deposit container. Specified as one of the following flags:

Value	Meaning
WFS_DEP_DEPOK	The deposit transport is in a good state.
WFS_DEP_DEPINOP	The deposit transport is inoperative due to a hardware failure or media jam.
WFS_DEP_DEPUNKNOWN	Due to a hardware error or other condition, the state of the deposit transport cannot be determined.
WFS_DEP_DEPNOTSUPP	The physical device has no deposit transport.

fwEnvSupply

Specifies the state of the envelope supply unit as one of the following flags:

Value	Meaning
WFS_DEP_ENVOK	The envelope supply unit is in a good state (and locked).
WFS_DEP_ENVLOW	The envelope supply unit is present but low.
WFS_DEP_ENVEMPTY	The envelope supply unit is present but empty. No envelopes can be dispensed.
WFS_DEP_ENVINOP	The envelope supply unit is in an inoperable state. No envelopes can be dispensed.
WFS_DEP_ENVMISSING	The envelope supply unit is missing.
WFS_DEP_ENVNOTSUPP	The physical device has no envelope supply.
WFS_DEP_ENVUNLOCKED	The envelope supply unit is unlocked.
WFS_DEP_ENVUNKNOWN	Due to a hardware error or other condition, the state of the envelope supply cannot be determined.

fwEnvDispenser

Specifies the state of the envelope dispenser. Specified as one of the following flags:

Value	Meaning
WFS_DEP_ENVOK	The envelope dispenser is present and in a good state.
WFS_DEP_ENVINOP	The envelope dispenser is present but in an inoperable state. No envelopes can be dispensed.
WFS_DEP_ENVUNKNOWN	Due to a hardware error or other condition, the state of the envelope dispenser cannot be determined.
WFS_DEP_ENVNOTSUPP	The physical device has no envelope dispenser.

fwPrinter

Specifies the state of the printer. Specified as one of the following flags:

Value	Meaning
WFS_DEP_PTROK	The printer is present and in a good state.
WFS_DEP_PTRINOP	The printer is inoperative.
WFS_DEP_PTRUNKNOWN	Due to a hardware error or other condition, the state of the printer cannot be determined.
WFS_DEP_PTRNOTSUPP	The physical device has no printer.

fwToner

Specifies the state of the toner (or ink) for the printer. Specified as one of the following flags:

Value	Meaning
WFS_DEP_TONERFULL	The toner cassette is full.
WFS_DEP_TONERLOW	The toner in the printer is low.
WFS_DEP_TONEROUT	The toner in the printer is empty.
WFS_DEP_TONERUNKNOWN	Due to a hardware error or other condition, the state of the toner for the printer cannot be determined.
WFS_DEP_TONERNOTSUPP	The physical device has no toner.

fwShutter

Specifies the state of the shutter or door. Specified as one of the following flags:

Value	Meaning
WFS_DEP_SHTCLOSED	The shutter is closed.
WFS_DEP_SHTOPEN	The shutter is open.
WFS_DEP_SHTJAMMED	The shutter is jammed.
WFS_DEP_SHTUNKNOWN	Due to a hardware error or other condition, the state of the shutter cannot be determined.
WFS_DEP_SHTNOTSUPP	The physical device has no shutter.

wNumOfDeposits

Specifies the number of envelopes or bags in the deposit container. This value is persistent, i.e. maintained through power failures, opens, closes and system resets.

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.

dwGuidLights [...]

Specifies the state of the guidance light indicators. A number of guidance light types are defined below. Vendor specific guidance lights are defined starting from the end of the array. The maximum guidance light index is WFS_DEP_GUIDLIGHTS_MAX.

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

Value	Meaning	Type
WFS_DEP_GUIDANCE_NOT_AVAILABLE	The status is not available.	A
WFS_DEP_GUIDANCE_OFF	The light is turned off.	A
WFS_DEP_GUIDANCE_SLOW_FLASH	The light is blinking slowly.	B
WFS_DEP_GUIDANCE_MEDIUM_FLASH	The light is blinking medium frequency.	B
WFS_DEP_GUIDANCE_QUICK_FLASH	The light is blinking quickly.	B
WFS_DEP_GUIDANCE_CONTINUOUS	The light is turned on continuous (steady).	B
WFS_DEP_GUIDANCE_RED	The light is red.	C
WFS_DEP_GUIDANCE_GREEN	The light is green.	C
WFS_DEP_GUIDANCE_YELLOW	The light is yellow.	C
WFS_DEP_GUIDANCE_BLUE	The light is blue.	C
WFS_DEP_GUIDANCE_CYAN	The light is cyan.	C
WFS_DEP_GUIDANCE_MAGENTA	The light is magenta.	C
WFS_DEP_GUIDANCE_WHITE	The light is white.	C

| WFS_DEP_GUIDANCE_ENTRY | The light is in the entry state. | D |
| WFS_DEP_GUIDANCE_EXIT | The light is in the exit state. | D |

dwGuidLights [*WFS_DEP_GUIDANCE_ENVDEPOSITORY*]
Specifies the state of the guidance light indicator on the envelope depository unit.

dwGuidLights [*WFS_DEP_GUIDANCE_ENVDISPENSER*]
Specifies the state of the guidance light indicator on the envelope dispenser unit.

fwDepositLocation
Specifies the location of the item deposited at the end of the last WFS_CMD_DEP_ENTRY command. Specified as one of the following flags:

Value	Meaning
WFS_DEP_DEPLOCNOTSUPP	Reporting the location of the last deposit is not supported.
WFS_DEP_DEPLOCUNKNOWN	Cannot determine the location of the last deposited item.
WFS_DEP_DEPLOCCONTAINER	The item is in the container.
WFS_DEP_DEPLOCTRANSPORT	The item is in the transport.
WFS_DEP_DEPLOCPRINTER	The item is in the printer.
WFS_DEP_DEPLOCSHUTTER	The item is at the shutter (available for removal).
WFS_DEP_DEPLOCNONE	No item was entered on the last WFS_CMD_DEP_ENTRY.
WFS_DEP_DEPLOCREMOVED	The item was removed.

For devices capable of identifying item location, WFS_DEP_DEPLOCNONE is returned when the status is queried before any call to WFS_CMD_DEP_ENTRY.

wDevicePosition
Specifies the device position. The device position value is independent of the *fwDevice* value, e.g. when the device position is reported as WFS_DEP_DEVICENOTINPOSITION, *fwDevice* can have any of the values defined above (including WFS_DEP_DEVONLINE or WFS_DEP_DEVOFFLINE). If the device is not in its normal operating position (i.e. WFS_DEP_DEVICEINPOSITION) then media may not be presented through the normal customer interface. This value is one of the following values:

Value	Meaning
WFS_DEP_DEVICEINPOSITION	The device is in its normal operating position, or is fixed in place and cannot be moved.
WFS_DEP_DEVICENOTINPOSITION	The device has been removed from its normal operating position.
WFS_DEP_DEVICEPOSUNKNOWN	Due to a hardware error or other condition, the position of the device cannot be determined.
WFS_DEP_DEVICEPOSNOTSUPP	The physical device does not have the capability of detecting the position.

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

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

Value	Meaning
WFS_DEP_AFMNOTSUPP	No anti-fraud module is available.
WFS_DEP_AFMOK	Anti-fraud module is in a good state and no foreign device is detected.
WFS_DEP_AFMINOP	Anti-fraud module is inoperable.
WFS_DEP_AFMDEVICEDETECTED	Anti-fraud module detected the presence of a foreign device.

<table>
<tr><td>WFS_DEP_AFMUNKNOWN</td><td>The state of the anti-fraud module cannot be determined.</td></tr>
</table>

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

Comments Applications which require or expect specific information to be present in the *lpszExtra* parameter may not be device or vendor-independent.

In the case where communications with the device has been lost, the *fwDevice* field will report WFS_DEP_DEVPOWEROFF when the device has been removed or WFS_DEP_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_DEP_CAPABILITIES

Description   This command is used to retrieve the capabilities of the Depository.

Input Param   None.

Output Param   LPWFSDEPCAPS lpCaps;

```
typedef struct _wfs_dep_caps
    {
    WORD                wClass;
    WORD                fwType;
    WORD                fwEnvSupply;
    BOOL                bDepTransport;
    BOOL                bPrinter;
    BOOL                bToner;
    BOOL                bShutter;
    BOOL                bPrintOnRetracts;
    WORD                fwRetractEnvelope;
    WORD                wMaxNumChars;
    WORD                fwCharSupport;
    LPSTR               lpszExtra;
    DWORD               dwGuidLights[WFS_DEP_GUIDLIGHTS_SIZE];
    BOOL                bPowerSaveControl;
    BOOL                bAntiFraudModule;
    LPDWORD             lpdwSynchronizableCommands;
    } WFSDEPCAPS, *LPWFSDEPCAPS;
```

wClass
Specifies the logical service class as WFS_SERVICE_CLASS_DEP.

fwType
Specifies the type of the depository device as a combination of the following flags:

Value	Meaning
WFS_DEP_TYPEENVELOPE	Depository accepts envelopes.
WFS_DEP_TYPEBAGDROP	Depository accepts bags.

fwEnvSupply
Defines what type of Envelope Supply Unit exists as one of the following flags:

Value	Meaning
WFS_DEP_ENVMOTORIZED	Envelope Supply can dispense envelopes.
WFS_DEP_ENVMANUAL	Envelope Supply is manual and must be unlocked to allow envelopes to be taken. The Service Event, WFS_SRVE_DEP_ENVTAKEN, ~~can not~~cannot be sent and the Execute Command, WFS_CMD_DEP_RETRACT ~~can not~~cannot be supported.
WFS_DEP_ENVNONE	No Envelope Supply or Envelope Supply is manual and envelopes can be taken at any time. The Service Event, WFS_SRVE_DEP_ENVTAKEN, ~~can not~~cannot be sent and the Execute Command, WFS_CMD_DEP_RETRACT ~~can not~~cannot be supported.

bDepTransport
Specifies whether a deposit transport mechanism is available.

bPrinter
Specifies whether a printer is available.

bToner
Specifies whether the printer has a toner (or ink) cassette.

bShutter
Specifies whether a deposit transport shutter is available.

bPrintOnRetracts
Specifies whether the device can print the string specified in the *lpszPrintData* or *lpszUNICODEPrintData* field of the WFS_CMD_DEP_RETRACT command on retracted envelopes.

fwRetractEnvelope
Specifies the ability of the envelope dispenser to retract envelopes as one of the following flags:

Value	Meaning
WFS_DEP_NORETRACT	The envelope dispenser does not have the capability to retract envelopes.
WFS_DEP_RETRACTDEP	Retracted envelopes are put in the deposit container.
WFS_DEP_RETRACTDISP	Retracted envelopes are retracted back to the envelope dispenser.

wMaxNumChars
Specifies the maximum number of characters that can be printed on the envelope.

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

Value	Meaning
WFS_DEP_ASCII	ASCII is supported for execute command data values.
WFS_DEP_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.

dwGuidLights [...]
Specifies which guidance lights are available. A number of guidance light types are defined below. Vendor specific guidance lights are defined starting from the end of the array. The maximum guidance light index is WFS_DEP_GUIDLIGHTS_MAX.

In addition to supporting specific flash rates and colors, some guidance lights also have the capability to show directional movement representing "entry" and "exit". The "entry" state gives the impression of leading a user to place media into the device. The "exit" state gives the impression of ejection from a device to a user and would be used for retrieving media from the device. The elements of this array are specified as a combination of the following flags and indicate all of the possible flash rates (type B), colors (type C) and directions (type D) that the guidance light indicator is capable of handling. If the guidance light indicator only supports one color then no value of type C is returned. If the guidance light indicator does not support direction then no value of type D is returned. A value of WFS_DEP_GUIDANCE_NOT_AVAILABLE indicates that the device has no guidance light indicator or the device controls the light directly with no application control possible.

Value	Meaning	Type
WFS_DEP_GUIDANCE_NOT_AVAILABLE	There is no guidance light control available at this position.	A
WFS_DEP_GUIDANCE_OFF	The light is turned off.	A
WFS_DEP_GUIDANCE_SLOW_FLASH	The light is blinking slowly.	B
WFS_DEP_GUIDANCE_MEDIUM_FLASH	The light is blinking medium frequency.	B
WFS_DEP_GUIDANCE_QUICK_FLASH	The light is blinking quickly.	B
WFS_DEP_GUIDANCE_CONTINUOUS	The light is turned on continuous (steady).	B
WFS_DEP_GUIDANCE_RED	The light is red.	C
WFS_DEP_GUIDANCE_GREEN	The light is green.	C
WFS_DEP_GUIDANCE_YELLOW	The light is yellow.	C
WFS_DEP_GUIDANCE_BLUE	The light is blue.	C

WFS_DEP_GUIDANCE_CYAN	The light is cyan.	C
WFS_DEP_GUIDANCE_MAGENTA	The light is magenta.	C
WFS_DEP_GUIDANCE_WHITE	The light is white.	C
WFS_DEP_GUIDANCE_ENTRY	The light can be in the entry state.	D
WFS_DEP_GUIDANCE_EXIT	The light can be in the exit state.	D

dwGuidLights [*WFS_DEP_GUIDANCE_ENVDEPOSITORY*]
Specifies whether the guidance light indicator on the envelope depository unit is available.

dwGuidLights [*WFS_DEP_GUIDANCE_ENVDISPENSER*]
Specifies whether the guidance light indicator on the envelope dispenser unit is available.

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

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.

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_DEP_ENTRY

Description This command starts the entry of an envelope and attempts to deposit it into the deposit container.

The WFS_EXEE_DEP_INSERTDEPOSIT event will be generated when the device is ready to accept the deposit.

A deposit is considered to be successful if an envelope is inserted and the shutter closes such that the customer no longer has access to it. This includes cases where the deposited envelope reaches the deposit container, becomes jammed before reaching the container, or cannot be returned to the customer.

If a successful deposit takes place, then this command will always complete with WFS_SUCCESS, and any errors detected during the operation will be returned by the WFS_EXEE_DEP_DEPOSITERROR event.

If a successful deposit causes the deposit bin to reach a high or full threshold, a WFS_USRE_DEP_DEPTHRESHOLD event will be sent.

A deposit is considered to be unsuccessful if an envelope is inserted, an error occurs, and the customer has the ability to access it. This includes cases where an envelope is returned to the user, or cases where it becomes jammed but the customer is still able to access it.

If an unsuccessful deposit takes place, then the command will always complete with an appropriate error code, and any errors detected during the operation will be returned by the WFS_EXEE_DEP_DEPOSITERROR event.

If the envelope is entered and then returned to the exit slot for removal by the customer, if and the deposit device is capable of this operation (either hardware capability or hardware problems such as a jam may prohibit the envelope from being returned)), a WFS_SRVE_DEP_ENVTAKEN will be sent when it is removed.

For example, if the envelope entered has an incorrect size and the deposit was unsuccessful, the envelope is returned to the exit slot for removal by the customer. If the envelope is returned to the customer for removal, the command will complete with WFS_ERR_DEP_ENVSIZE. A WFS_SRVE_DEP_ENVTAKEN is sent when the envelope is removed. But if returning the envelope is not possible and the customer cannot access the envelope, the command will complete with WFS_SUCCESS and a WFS_EXEE_DEP_DEPOSITERROR event is sent reporting a WFS_ERR_DEP_ENVSIZE.

Input Param LPWFSDEPENVELOPE lpEnvelope;

```
typedef struct _wfs_dep_envelope
    {
    LPSTR               lpszPrintData;
    LPWSTR              lpszUNICODEPrintData;
    } WFSDEPENVELOPE, *LPWFSDEPENVELOPE;
```

lpszPrintData
Specifies the data that will be printed on the envelope that is entered by the customer.

lpszUNICODEPrintData
Specifies the UNICODE data that will be printed on the envelope that is entered by the customer.

The *lpszUNICODEPrintData* field should only be used if the Service Provider supports UNICODE. The *lpszPrintData* and *lpszUNICODEPrintData* 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_DEP_DEPFULL	The deposit container is full.

WFS_ERR_DEP_DEPJAMMED   An envelope jam occurred in the deposit transport between the entry slot and the deposit container.

WFS_ERR_DEP_ENVSIZE	The envelope entered has an incorrect size.
WFS_ERR_DEP_PTRFAIL	The printer failed.
WFS_ERR_DEP_SHTNOTCLOSED	The shutter failed to close.
WFS_ERR_DEP_SHTNOTOPENED	The shutter failed to open.
WFS_ERR_DEP_CONTMISSING	The deposit container is not present.
WFS_ERR_DEP_DEPUNKNOWN	The result of the deposit is not known.
WFS_ERR_DEP_CHARSETNOTSUPP	Character set(s) supported by Service Provider is inconsistent with use of *lpszPrintData* or *lpszUNICODEPrintData* fields.
WFS_ERR_DEP_TONEROUT	Toner or ink supply is empty or printing contrast with ribbon is not sufficient. This error can only occur when a print string was passed in the input parameter.

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

Value	Meaning
WFS_SRVE_DEP_ENVTAKEN	The envelope has been taken by the user.
WFS_EXEE_DEP_ENVDEPOSITED	The envelope has been deposited in the deposit container.
WFS_EXEE_DEP_DEPOSITERROR	An error occurred during the deposit operation.
WFS_USRE_DEP_DEPTHRESHOLD	This user event is used to specify that the state of the deposit container reached a threshold.
WFS_USRE_DEP_TONERTHRESHOLD	This user event is used to specify that the state of the toner supply reached a threshold.
WFS_SRVE_DEP_ENVINSERTED	An envelope has been inserted by the user.
WFS_EXEE_DEP_INSERTDEPOSIT	Device is ready to accept deposit from the user.

Comments   If the data specified in *lpszPrintData* or *lpszUNICODEPrintData* is longer than the maximum allowed characters, the error code WFS_ERR_INVALID_DATA will be returned.

5.2   WFS_CMD_DEP_DISPENSE

Description   This command is used to dispense an envelope from the envelope supply. This command will either action the dispensing of an envelope from the envelope supply or will unlock the envelope supply for manual access.

Input Param   None.

Output Param   None.

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

Value	Meaning
WFS_ERR_DEP_ENVEMPTY	There is no envelope in the envelope unit.
WFS_ERR_DEP_ENVJAMMED	An envelope jam occurred in the dispenser transport between the envelope supply and the output slot.
WFS_ERR_DEP_SHTNOTOPENED	The shutter failed to open.

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

Value	Meaning
WFS_SRVE_DEP_ENVTAKEN	The envelope has been taken by the user.
WFS_USRE_DEP_ENVTHRESHOLD	This user event is used to specify that the state of the envelope supply reached a threshold.

Comments   None.

5.3   WFS_CMD_DEP_RETRACT

Description   This command is used to retract an envelope that was not taken by a customer after an envelope dispense operation. The given string is printed on the envelope and the envelope is retracted into the deposit container or back to the envelope dispenser, depending on the capabilities of the physical device. If a retract to the deposit bin causes the deposit bin to reach a high or full threshold, a WFS_USRE_DEP_DEPTHRESHOLD event will be sent.

This command will only return with an error code if the retract has not taken place. The error code will then describe the reason for the failure.

Input Param   LPWFSDEPENVELOPE lpEnvelope;

```
typedef struct _wfs_dep_envelope
    {
    LPSTR                lpszPrintData;
    LPSTR                lpszUNICODEPrintData;
    } WFSDEPENVELOPE, *LPWFSDEPENVELOPE;
```

lpszPrintData
Specifies the data that will be printed on the envelope that is retracted.

lpszUNICODEPrintData
Specifies the UNICODE data that will be printed on the envelope that is retracted.

The *lpszUNICODEPrintData* field should only be used if the Service Provider supports UNICODE. The *lpszPrintData* and *lpszUNICODEPrintData* 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_DEP_DEPFULL	The deposit container is full.
WFS_ERR_DEP_DEPJAMMED	An envelope jam occurred in the deposit transport between the entry slot and the deposit container (may only occur with hardware that retracts to the deposit container).
WFS_ERR_DEP_ENVJAMMED	An envelope jam occurred between the entry slot and the envelope container (may only occur with hardware that retracts to the envelope container).
WFS_ERR_DEP_NOENV	No envelope to retract.
WFS_ERR_DEP_PTRFAIL	The printer failed.
WFS_ERR_DEP_SHTNOTCLOSED	The shutter failed to close.
WFS_ERR_DEP_CONTMISSING	The deposit container is not present.
WFS_ERR_DEP_CHARSETNOTSUPP	Character set(s) supported by Service Provider is inconsistent with use of *lpszPrintData* or *lpszUNICODEPrintData* fields.
WFS_ERR_DEP_TONEROUT	Toner or ink supply is empty or printing contrast with ribbon is not sufficient.

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

Value	Meaning
WFS_USRE_DEP_DEPTHRESHOLD	This user event is used to specify that the state of the deposit container reached a threshold.
WFS_USRE_DEP_TONERTHRESHOLD	This user event is used to specify that the state of the toner supply reached a threshold.
WFS_SRVE_DEP_ENVTAKEN	The envelope has been taken by the user.

Comments   If the data specified in *lpszPrintData* or *lpszUNICODEPrintData* is longer than the maximum allowed characters, the error code WFS_ERR_INVALID_DATA will be returned.

5.4 WFS_CMD_DEP_RESET_COUNT

Description   This command is used to reset the present value for number of envelopes/bags in the deposit container to zero.

Input Param   None.

Output Param   None.

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

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

Value	Meaning
WFS_USRE_DEP_DEPTHRESHOLD	This user event is used to specify that the state of the deposit container reached a threshold.

Comments   None.

5.5   WFS_CMD_DEP_RESET

Description   Sends a service reset to the Service Provider. The Service Provider may reset the deposit device and also the envelope dispenser, if possible. Any media found in the device can be either captured or completely ejected (depending on hardware). If a capture into the deposit bin causes the deposit bin to reach a high or full threshold, a WFS_USRE_DEP_DEPTHRESHOLD event will be sent. If the WFS_CMD_DEP_RESET command is requested to eject the media and the hardware is not capable of this operation either due to hardware capability or hardware error such as a jam, the Service Provider will retract the media in order to attempt to make the device operational. The WFS_SRVE_DEP_MEDIADETECTED event will indicate the position of the detected media following completion of the command. If the input parameter to the WFS_CMD_DEP_RESET command is NULL, the Service Provider will go through default actions to clear the deposit transport. The WFS_SRVE_DEP_MEDIADETECTED event will indicate the position of any detected media following completion of the command. The envelope dispenser will go through the most effective means to clear any jammed media.

Input Param   LPDWORD lpdwDepMediaControl;

Specifies the action that should be done if deposited media is detected during the reset operation, as one of the following values:

Value	Meaning
WFS_DEP_CTRLEJECT	Any media detected in the device should be completed ejected (depending on the hardware).
WFS_DEP_CTRLRETRACT	Any media detected in the device should be deposited into the deposit container during the reset operation.

If *lpdwDepMediaControl* is set to NULL, the Service Provider will go through default actions to clear the deposit transport.

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_DEP_DEPFULL	The deposit container is full.
WFS_ERR_DEP_DEPJAMMED	An envelope jam occurred in the deposit transport.
WFS_ERR_DEP_ENVJAMMED	An envelope jam occurred in the dispenser transport between the envelope supply and the output slot.
WFS_ERR_DEP_SHTNOTOPENED	The shutter failed to open.
WFS_ERR_DEP_SHTNOTCLOSED	The shutter failed to close.
WFS_ERR_DEP_CONTMISSING	The deposit container is not present.

Events   In addition to the generic events defined in [Ref. 1], the following events may be generated by this command, if the appropriate situation occurs and the device service has the capability to detect the situation:

Value	Meaning
WFS_SRVE_DEP_ENVTAKEN	The envelope has been taken by the user.
WFS_USRE_DEP_DEPTHRESHOLD	This user event is used to specify that the state of the deposit container reached a threshold.
WFS_SRVE_DEP_MEDIADETECTED	Media is detected in the device during a reset operation.

Comments   This command is used by an application control program to cause a device to reset itself to a known good condition. Persistent values may change, but will not be reset as a result of this command (i.e. if an envelope is captured, the *wNumOfDeposits* value in the WFSDEPSTATUS structure will be incremented, but never reset to zero).

5.6   WFS_CMD_DEP_SET_GUIDANCE_LIGHT

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

Input Param   LPWFSDEPSETGUIDLIGHT lpSetGuidLight;

```
typedef struct _wfs_dep_set_guidlight
    {
    WORD                wGuidLight;
    DWORD               dwCommand;
    } WFSDEPSETGUIDLIGHT, *LPWFSDEPSETGUIDLIGHT;
```

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

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

Value	Meaning	Type
WFS_DEP_GUIDANCE_OFF	The light indicator is turned off.	A
WFS_DEP_GUIDANCE_SLOW_FLASH	The light indicator is set to flash slowly.	B
WFS_DEP_GUIDANCE_MEDIUM_FLASH	The light indicator is set to flash medium frequency.	B
WFS_DEP_GUIDANCE_QUICK_FLASH	The light indicator is set to flash quickly.	B
WFS_DEP_GUIDANCE_CONTINUOUS	The light indicator is turned on continuously (steady).	B
WFS_DEP_GUIDANCE_RED	The light indicator color is set to red.	C
WFS_DEP_GUIDANCE_GREEN	The light indicator color is set to green.	C
WFS_DEP_GUIDANCE_YELLOW	The light indicator color is set to yellow.	C
WFS_DEP_GUIDANCE_BLUE	The light indicator color is set to blue.	C
WFS_DEP_GUIDANCE_CYAN	The light indicator color is set to cyan.	C
WFS_DEP_GUIDANCE_MAGENTA	The light indicator color is set to magenta.	C
WFS_DEP_GUIDANCE_WHITE	The light indicator color is set to white.	C
WFS_DEP_GUIDANCE_ENTRY	The light indicator is set to the entry state.	D
WFS_DEP_GUIDANCE_EXIT	The light indicator is set to the exit state.	D

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_DEP_INVALID_PORT	An attempt to set a guidance light to a new value was invalid because the guidance light does not exist.

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

Comments   Guidance light support was added into the DEP primarily to support guidance lights for

workstations where more than one instance of a DEP is present. The original SIU guidance light mechanism was not able to manage guidance lights for workstations with multiple DEPs. This command can also be used to set the status of the DEP guidance lights when only one instance of a DEP is present.

The slow and medium flash rates must not be greater than 2.0 Hz. It should be noted that in order to comply with American Disabilities Act guidelines only a slow or medium flash rate must be used.

5.7   WFS_CMD_DEP_SUPPLY_REPLENISH

Description   After the supplies have been replenished, this command is used to indicate that the specified supplies have been replenished and are expected to be in a healthy state.

Hardware that cannot detect the level of a supply and reports on the supply's status using metrics (or some other means), must assume the supply has been fully replenished after this command is issued. The appropriate threshold event must be broadcast.

Hardware that can detect the level of a supply must update its status based on its sensors, generate a threshold event if appropriate and succeed the command even if the supply has not been replenished. If it has already detected the level and reported the threshold before this command was issued, the command must succeed and no threshold event is required.

Input Param   LPWFSDEPSUPPLYREPLEN lpSupplyReplen;

```
typedef struct _wfs_dep_supply_replen
    {
    WORD                fwSupplyReplen;
    } WFSDEPSUPPLYREPLEN, *LPWFSDEPSUPPLYREPLEN;
```

fwSupplyReplen
Specifies the supply that was replenished as a combination of the following values:

Value	Meaning
WFS_DEP_REPLEN_ENV	The envelope supply was replenished.
WFS_DEP_REPLEN_TONER	The toner supply was replenished.

Output Param   None.

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

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

Value	Meaning
WFS_USRE_DEP_ENVTHRESHOLD	This user event is used to specify that the state of the envelope supply threshold has been cleared.
WFS_USRE_DEP_TONERTHRESHOLD	This user event is used to specify that the state of the toner (or ink supply or the state of a ribbon) supply threshold has been cleared.

Comments   If any one of the specified supplies is not supported by a Service Provider, WFS_ERR_UNSUPP_DATA should be returned, and no replenishment actions will be taken by the Service Provider.

5.8 WFS_CMD_DEP_POWER_SAVE_CONTROL

Description   This command activates or deactivates the power-saving mode.

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

Input Param   LPWFSDEPPOWERSAVECONTROL lpPowerSaveControl;

```
typedef struct _wfs_dep_power_save_control
    {
    USHORT              usMaxPowerSaveRecoveryTime;
    } WFSDEPPOWERSAVECONTROL, *LPWFSDEPPOWERSAVECONTROL;
```

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

Output Param   None.

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

Value	Meaning
WFS_ERR_DEP_POWERSAVETOOSHORT	The power saving mode has not been activated because the device is not able to resume from the power saving mode within the specified *usMaxPowerSaveRecoveryTime* value.
WFS_ERR_DEP_POWERSAVEMEDIAPRESENT	
	The power saving mode has not been activated because media is present inside the device.

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

Value	Meaning
WFS_SRVE_DEP_POWER_SAVE_CHANGE	The power save recovery time has changed.

Comments   None.

5.9 WFS_CMD_DEP_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_DEP_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_DEP_SYNCHRONIZE_COMMAND again in order to start a synchronization.

Input Param   LPWFSDEPSYNCHRONIZECOMMAND lpSynchronizeCommand;

```
typedef struct _wfs_dep_synchronize_command
    {
    DWORD                       dwCommand;
    LPVOID                      lpCmdData;
    } WFSDEPSYNCHRONIZECOMMAND, *LPWFSDEPSYNCHRONIZECOMMAND;
```

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_DEP_ENTRY then *lpCmdData* will point to a WFSDEPENVELOPE 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_DEP_COMMANDUNSUPP	The command specified in the *dwCommand* field is not supported by the Service Provider.
WFS_ERR_DEP_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   For sample flows of this synchronization see the [Ref 1] Appendix C.

6. Events

6.1 WFS_SRVE_DEP_ENVTAKEN

Description   This service event is used to specify that the envelope has been taken by the customer.

Event Param   None.

Comments   None.

6.2 WFS_EXEE_DEP_ENVDEPOSITED

Description   This execute event is used to specify that the envelope has been deposited in the deposit container.

Event Param   None.

Comments   None.

6.3   WFS_EXEE_DEP_DEPOSITERROR

Description   This execute event is used to specify that an error occurred during the deposit operation. For every error that occurred a single execute event is generated.

Event Param   LPLONG lplError;

lplError
Pointer to the error code. For a list of possible error conditions see the description of the WFS_CMD_DEP_ENTRY command.

Comments   None.

6.4   WFS_USRE_DEP_DEPTHRESHOLD

Description   This user event is used to specify that the state of the deposit container reached a threshold.

Event Param   LPWORD lpwDepositThreshold;

lpwDepositThreshold
~~Specified as~~Pointer to one of the following flags:

Value	Meaning
WFS_DEP_DEPOK	The deposit container is in a good state.
WFS_DEP_DEPHIGH	The deposit container is almost full (threshold).
WFS_DEP_DEPFULL	The deposit container is full.

Comments   None.

6.5   WFS_USRE_DEP_TONERTHRESHOLD

Description   This user event is used to specify that the state of the toner (or ink supply or the state of a ribbon) reached a threshold.

Event Param   LPWORD lpwTonerThreshold;

lpwTonerThreshold
~~Specified as~~Pointer to one of the following flags:

Value	Meaning
WFS_DEP_TONERFULL	The toner or ink supply is full or the ribbon is OK.
WFS_DEP_TONERLOW	The toner or ink supply is low or the print contrast with a ribbon is weak.
WFS_DEP_TONEROUT	The toner or ink supply is empty or the print contrast with a ribbon is not sufficient any more.

Comments   None.

6.6   WFS_USRE_DEP_ENVTHRESHOLD

Description   This user event is used to specify that the state of the envelope supply reached a threshold.

Event Param   LPWORD lpwEnvelopeThreshold;

lpwEnvelopeThreshold
~~Specified as~~Pointer to one of the following flags:

Value	Meaning
WFS_DEP_ENVOK	The envelope supply is present and in a good state.
WFS_DEP_ENVLOW	The envelope supply is present but low.
WFS_DEP_ENVEMPTY	The envelope supply is present but empty. No envelopes can be dispensed.

Comments   None.

6.7   WFS_SRVE_DEP_CONTINSERTED

Description   This service event is used to specify that the deposit container has been reinserted into the device.

Event Param   None.

Comments   None.

6.8   WFS_SRVE_DEP_CONTREMOVED

Description This service event is used to specify that the deposit container has been removed from the device.

Event Param None.

Comments None.

6.9   WFS_SRVE_DEP_ENVINSERTED

Description   This service event is used to specify that an envelope has been inserted by the customer.

Event Param   None.

Comments   None.

6.10 WFS_SRVE_DEP_MEDIADETECTED

Description   This event is generated when media is detected in the device during a reset operation. The media may be detected as a result of the reset operation on the envelope dispenser, the envelope depositor, or both.

Event Param   LPWFSDEPMEDIADETECTED lpMediaDetected;

```
typedef struct _wfs_dep_media_detected
    {
    WORD                wDispenseMedia;
    WORD                wDepositMedia;
    } WFSDEPMEDIADETECTED, *LPWFSDEPMEDIADETECTED;
```

wDispenseMedia
Specifies the dispensed envelope position after the reset operation, as one of the following values:

Value	Meaning
WFS_DEP_NOMEDIA	No dispensed media was detected during the reset operation.
WFS_DEP_MEDIARETRACTED	The media was retracted into the deposit container during the reset operation.
WFS_DEP_MEDIADISPENSER	The media was retracted into the envelope dispenser during the reset operation.
WFS_DEP_MEDIAEJECTED	The media is in the exit slot.
WFS_DEP_MEDIAJAMMED	The media is jammed in the device.
WFS_DEP_MEDIAUNKNOWN	The media is in an unknown position.

wDepositMedia
Specifies the deposited media position after the reset operation, as one of the following values:

Value	Meaning
WFS_DEP_NOMEDIA	No deposited media was detected during the reset operation.
WFS_DEP_MEDIARETRACTED	The media was retracted into the deposit container during the reset operation.
WFS_DEP_MEDIAEJECTED	The media is in the exit slot.
WFS_DEP_MEDIAJAMMED	The media is jammed in the device.
WFS_DEP_MEDIAUNKNOWN	The media is in an unknown position.

Comments   None.

6.11 WFS_EXEE_DEP_INSERTDEPOSIT

Description   This event notifies the application when the device is ready for the user to make the deposit. This event is mandatory.

Event Param   None.

Comments   None.

6.12 WFS_SRVE_DEP_DEVICEPOSITION

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

Event Param   LPWFSDEPDEVICEPOSITION lpDevicePosition;

```
typedef struct _wfs_dep_device_position
    {
    WORD                    wPosition;
    } WFSDEPDEVICEPOSITION, *LPWFSDEPDEVICEPOSITION;
```

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

Value	Meaning
WFS_DEP_DEVICEINPOSITION	The device is in its normal operating position.
WFS_DEP_DEVICENOTINPOSITION	The device has been removed from its normal operating position.
WFS_DEP_DEVICEPOSUNKNOWN	The position of the device cannot be determined.

Comments   None.

6.13 WFS_SRVE_DEP_POWER_SAVE_CHANGE

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

Event Param   LPWFSDEPPOWERSAVECHANGE lpPowerSaveChange;

```
typedef struct _wfs_dep_power_save_change
    {
    USHORT                  usPowerSaveRecoveryTime;
    } WFSDEPPOWERSAVECHANGE, *LPWFSDEPPOWERSAVECHANGE;
```

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

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

7. C - Header file

```
/***************************************************************************
*                                                                         *
* xfsdep.h   XFS - Depository (DEP) definitions                           *
*                                                                         *
*             Version 3.40   (December 6 2019) 50   (November 18 2022)    *
*                                                                         *
*                                                                         *
***************************************************************************/

#ifndef __INC_XFSDEP__H
#define __INC_XFSDEP__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

/* be aware of alignment */
#pragma pack(push,1)

/* values of WFSDEPCAPS.wClass */

#define      WFS_SERVICE_CLASS_DEP                 (6)
#define      WFS_SERVICE_CLASS_VERSION_DEP         (0x28030x3203) /* Version 3.4050 */
#define      WFS_SERVICE_CLASS_NAME_DEP            "DEP"

#define      DEP_SERVICE_OFFSET                    (WFS_SERVICE_CLASS_DEP * 100)

/* DEP Info Commands */

#define      WFS_INF_DEP_STATUS                    (DEP_SERVICE_OFFSET + 1)
#define      WFS_INF_DEP_CAPABILITIES              (DEP_SERVICE_OFFSET + 2)

/* DEP Execute Commands */

#define      WFS_CMD_DEP_ENTRY                     (DEP_SERVICE_OFFSET + 1)
#define      WFS_CMD_DEP_DISPENSE                  (DEP_SERVICE_OFFSET + 2)
#define      WFS_CMD_DEP_RETRACT                   (DEP_SERVICE_OFFSET + 3)
#define      WFS_CMD_DEP_RESET_COUNT               (DEP_SERVICE_OFFSET + 5)
#define      WFS_CMD_DEP_RESET                     (DEP_SERVICE_OFFSET + 6)
#define      WFS_CMD_DEP_SET_GUIDANCE_LIGHT        (DEP_SERVICE_OFFSET + 7)
#define      WFS_CMD_DEP_SUPPLY_REPLENISH          (DEP_SERVICE_OFFSET + 8)
#define      WFS_CMD_DEP_POWER_SAVE_CONTROL        (DEP_SERVICE_OFFSET + 9)
#define      WFS_CMD_DEP_SYNCHRONIZE_COMMAND       (DEP_SERVICE_OFFSET + 10)

/* DEP Messages */

#define      WFS_SRVE_DEP_ENVTAKEN                 (DEP_SERVICE_OFFSET + 1)
#define      WFS_EXEE_DEP_ENVDEPOSITED             (DEP_SERVICE_OFFSET + 2)
#define      WFS_EXEE_DEP_DEPOSITERROR             (DEP_SERVICE_OFFSET + 3)
#define      WFS_USRE_DEP_DEPTHRESHOLD             (DEP_SERVICE_OFFSET + 4)
#define      WFS_USRE_DEP_TONERTHRESHOLD           (DEP_SERVICE_OFFSET + 5)
#define      WFS_USRE_DEP_ENVTHRESHOLD             (DEP_SERVICE_OFFSET + 6)
#define      WFS_SRVE_DEP_CONTINSERTED             (DEP_SERVICE_OFFSET + 7)
#define      WFS_SRVE_DEP_CONTREMOVED              (DEP_SERVICE_OFFSET + 8)
#define      WFS_SRVE_DEP_ENVINSERTED              (DEP_SERVICE_OFFSET + 9)
#define      WFS_SRVE_DEP_MEDIADETECTED            (DEP_SERVICE_OFFSET + 10)
#define      WFS_EXEE_DEP_INSERTDEPOSIT            (DEP_SERVICE_OFFSET + 11)
#define      WFS_SRVE_DEP_DEVICEPOSITION           (DEP_SERVICE_OFFSET + 12)
#define      WFS_SRVE_DEP_POWER_SAVE_CHANGE        (DEP_SERVICE_OFFSET + 13)

/* values of WFSDEPSTATUS.fwDevice */

#define      WFS_DEP_DEVONLINE                     WFS_STAT_DEVONLINE
#define      WFS_DEP_DEVOFFLINE                    WFS_STAT_DEVOFFLINE
#define      WFS_DEP_DEVPOWEROFF                   WFS_STAT_DEVPOWEROFF
```

```
#define       WFS_DEP_DEVBUSY                     WFS_STAT_DEVBUSY
#define       WFS_DEP_DEVNODEVICE                 WFS_STAT_DEVNODEVICE
#define       WFS_DEP_DEVHWERROR                  WFS_STAT_DEVHWERROR
#define       WFS_DEP_DEVUSERERROR                WFS_STAT_DEVUSERERROR
#define       WFS_DEP_DEVFRAUDATTEMPT             WFS_STAT_DEVFRAUDATTEMPT
#define       WFS_DEP_DEVPOTENTIALFRAUD           WFS_STAT_DEVPOTENTIALFRAUD

/* values of WFSDEPSTATUS.fwDepContainer, fwDepTransport */

#define       WFS_DEP_DEPOK                 (0)
#define       WFS_DEP_DEPHIGH               (1)
#define       WFS_DEP_DEPFULL               (2)
#define       WFS_DEP_DEPINOP               (3)
#define       WFS_DEP_DEPMISSING            (4)
#define       WFS_DEP_DEPUNKNOWN            (5)
#define       WFS_DEP_DEPNOTSUPP            (6)

/* values of WFSDEPSTATUS.fwEnvSupply, fwEnvDispenser */

#define       WFS_DEP_ENVOK                 (0)
#define       WFS_DEP_ENVLOW                (1)
#define       WFS_DEP_ENVEMPTY              (2)
#define       WFS_DEP_ENVINOP               (3)
#define       WFS_DEP_ENVMISSING            (4)
#define       WFS_DEP_ENVUNKNOWN            (5)
#define       WFS_DEP_ENVNOTSUPP            (6)
#define       WFS_DEP_ENVUNLOCKED           (7)

/* values of WFSDEPSTATUS.fwPrinter */

#define       WFS_DEP_PTROK                 (0)
#define       WFS_DEP_PTRINOP               (1)
#define       WFS_DEP_PTRUNKNOWN            (2)
#define       WFS_DEP_PTRNOTSUPP            (3)

/* values of WFSDEPSTATUS.fwToner */

#define       WFS_DEP_TONERFULL             (0)
#define       WFS_DEP_TONERLOW              (1)
#define       WFS_DEP_TONEROUT              (2)
#define       WFS_DEP_TONERUNKNOWN          (3)
#define       WFS_DEP_TONERNOTSUPP          (4)

/* values of WFSDEPSTATUS.fwShutter */

#define       WFS_DEP_SHTCLOSED             (0)
#define       WFS_DEP_SHTOPEN               (1)
#define       WFS_DEP_SHTJAMMED             (2)
#define       WFS_DEP_SHTUNKNOWN            (3)
#define       WFS_DEP_SHTNOTSUPP            (4)

/* Size and max index of dwGuidLights array */

#define       WFS_DEP_GUIDLIGHTS_SIZE       (32)
#define       WFS_DEP_GUIDLIGHTS_MAX        (WFS_DEP_GUIDLIGHTS_SIZE - 1)

/* Indices of WFSDEPSTATUS.dwGuidLights [...]
              WFSDEPCAPS.dwGuidLights [...]
*/

#define       WFS_DEP_GUIDANCE_ENVDEPOSITORY      (0)
#define       WFS_DEP_GUIDANCE_ENVDISPENSER       (1)

/* Values of WFSDEPSTATUS.dwGuidLights [...]
              WFSDEPCAPS.dwGuidLights [...] */

#define       WFS_DEP_GUIDANCE_NOT_AVAILABLE      (0x00000000)
#define       WFS_DEP_GUIDANCE_OFF                (0x00000001)
#define       WFS_DEP_GUIDANCE_SLOW_FLASH         (0x00000004)
#define       WFS_DEP_GUIDANCE_MEDIUM_FLASH       (0x00000008)
```

```
#define      WFS_DEP_GUIDANCE_QUICK_FLASH        (0x00000010)
#define      WFS_DEP_GUIDANCE_CONTINUOUS         (0x00000080)
#define      WFS_DEP_GUIDANCE_RED                (0x00000100)
#define      WFS_DEP_GUIDANCE_GREEN              (0x00000200)
#define      WFS_DEP_GUIDANCE_YELLOW             (0x00000400)
#define      WFS_DEP_GUIDANCE_BLUE               (0x00000800)
#define      WFS_DEP_GUIDANCE_CYAN               (0x00001000)
#define      WFS_DEP_GUIDANCE_MAGENTA            (0x00002000)
#define      WFS_DEP_GUIDANCE_WHITE              (0x00004000)
#define      WFS_DEP_GUIDANCE_ENTRY              (0x00100000)
#define      WFS_DEP_GUIDANCE_EXIT               (0x00200000)


/* values of WFSDEPSTATUS.fwDepositLocation */

#define WFS_DEP_DEPLOCNOTSUPP                (0)
#define WFS_DEP_DEPLOCUNKNOWN                (1)
#define WFS_DEP_DEPLOCCONTAINER             (2)
#define WFS_DEP_DEPLOCTRANSPORT             (3)
#define WFS_DEP_DEPLOCPRINTER               (4)
#define WFS_DEP_DEPLOCSHUTTER               (5)
#define WFS_DEP_DEPLOCNONE                  (6)
#define WFS_DEP_DEPLOCREMOVED               (7)


/* values of WFSDEPSTATUS.wDevicePosition
            WFSDEPDEVICEPOSITION.wPosition */

#define      WFS_DEP_DEVICEINPOSITION       (0)
#define      WFS_DEP_DEVICENOTINPOSITION    (1)
#define      WFS_DEP_DEVICEPOSUNKNOWN       (2)
#define      WFS_DEP_DEVICEPOSNOTSUPP       (3)

/* values of WFSDEPCAPS.fwType */

#define      WFS_DEP_TYPEENVELOPE       (0x0001)
#define      WFS_DEP_TYPEBAGDROP        (0x0002)

/* values of WFSDEPCAPS.fwEnvSupply */

#define      WFS_DEP_ENVMOTORIZED       (1)
#define      WFS_DEP_ENVMANUAL          (2)
#define      WFS_DEP_ENVNONE            (3)

/* values of WFSDEPCAPS.fwRetractEnvelope */

#define      WFS_DEP_NORETRACT          (1)
#define      WFS_DEP_RETRACTDEP         (2)
#define      WFS_DEP_RETRACTDISP        (3)

/* values of WFSDEPCAPS.fwCharSupport, WFSDEPENVELOPE.fwCharSupport */

#define      WFS_DEP_ASCII                 (0x0001)
#define      WFS_DEP_UNICODE               (0x0002)

/* values of dwDepMediaControl  */

#define      WFS_DEP_CTRLEJECT             (0x0001)
#define      WFS_DEP_CTRLRETRACT           (0x0002)


/* values of WFSDEPMEDIADETECTED.wDispenseMedia, wDepositMedia */

#define      WFS_DEP_NOMEDIA               (1)
#define      WFS_DEP_MEDIARETRACTED        (2)
#define      WFS_DEP_MEDIADISPENSER        (3)
#define      WFS_DEP_MEDIAEJECTED          (4)
#define      WFS_DEP_MEDIAJAMMED           (5)
#define      WFS_DEP_MEDIAUNKNOWN          (6)

/* values of WFSDEPSUPPLYREPLEN.fwSupplyReplen */
```

```
#define    WFS_DEP_REPLEN_ENV                    (0x0001)
#define    WFS_DEP_REPLEN_TONER                  (0x0002)

/* values of WFSDEPSTATUS.wAntiFraudModule */

#define    WFS_DEP_AFMNOTSUPP                    (0)
#define    WFS_DEP_AFMOK                         (1)
#define    WFS_DEP_AFMINOP                       (2)
#define    WFS_DEP_AFMDEVICEDETECTED             (3)
#define    WFS_DEP_AFMUNKNOWN                    (4)

#define WFS_ERR_DEP_DEPFULL                      (-(DEP_SERVICE_OFFSET + 0))
#define WFS_ERR_DEP_DEPJAMMED                    (-(DEP_SERVICE_OFFSET + 1))
#define WFS_ERR_DEP_ENVEMPTY                     (-(DEP_SERVICE_OFFSET + 2))
#define WFS_ERR_DEP_ENVJAMMED                    (-(DEP_SERVICE_OFFSET + 3))
#define WFS_ERR_DEP_ENVSIZE                      (-(DEP_SERVICE_OFFSET + 4))
#define WFS_ERR_DEP_NOENV                        (-(DEP_SERVICE_OFFSET + 5))
#define WFS_ERR_DEP_PTRFAIL                      (-(DEP_SERVICE_OFFSET + 6))
#define WFS_ERR_DEP_SHTNOTCLOSED                 (-(DEP_SERVICE_OFFSET + 7))
#define WFS_ERR_DEP_SHTNOTOPENED                 (-(DEP_SERVICE_OFFSET + 8))
#define WFS_ERR_DEP_CONTMISSING                  (-(DEP_SERVICE_OFFSET + 9))
#define WFS_ERR_DEP_DEPUNKNOWN                   (-(DEP_SERVICE_OFFSET + 10))
#define WFS_ERR_DEP_CHARSETNOTSUPP               (-(DEP_SERVICE_OFFSET + 11))
#define WFS_ERR_DEP_TONEROUT                     (-(DEP_SERVICE_OFFSET + 12))
#define WFS_ERR_DEP_INVALID_PORT                 (-(DEP_SERVICE_OFFSET + 13))
#define WFS_ERR_DEP_POWERSAVETOOSHORT            (-(DEP_SERVICE_OFFSET + 14))
#define WFS_ERR_DEP_POWERSAVEMEDIAPRESENT        (-(DEP_SERVICE_OFFSET + 15))
#define WFS_ERR_DEP_COMMANDUNSUPP                (-(DEP_SERVICE_OFFSET + 16))
#define WFS_ERR_DEP_SYNCHRONIZEUNSUPP            (-(DEP_SERVICE_OFFSET + 17))

/*=================================================================*/
/* DEP Info Command Structures and variables */
/*=================================================================*/

typedef struct _wfs_dep_status
{
    WORD          fwDevice;
    WORD          fwDepContainer;
    WORD          fwDepTransport;
    WORD          fwEnvSupply;
    WORD          fwEnvDispenser;
    WORD          fwPrinter;
    WORD          fwToner;
    WORD          fwShutter;
    WORD          wNumOfDeposits;
    LPSTR         lpszExtra;
    DWORD         dwGuidLights[WFS_DEP_GUIDLIGHTS_SIZE];
    WORD          fwDepositLocation;
    WORD          wDevicePosition;
    USHORT        usPowerSaveRecoveryTime;
    WORD          wAntiFraudModule;
} WFSDEPSTATUS, *LPWFSDEPSTATUS;

typedef struct _wfs_dep_caps
{
    WORD          wClass;
    WORD          fwType;
    WORD          fwEnvSupply;
    BOOL          bDepTransport;
    BOOL          bPrinter;
    BOOL          bToner;
    BOOL          bShutter;
    BOOL          bPrintOnRetracts;
    WORD          fwRetractEnvelope;
    WORD          wMaxNumChars;
    WORD          fwCharSupport;
    LPSTR         lpszExtra;
    DWORD         dwGuidLights[WFS_DEP_GUIDLIGHTS_SIZE];
    BOOL          bPowerSaveControl;
```

```
    BOOL            bAntiFraudModule;
    LPDWORD         lpdwSynchronizableCommands;
} WFSDEPCAPS, *LPWFSDEPCAPS;


/*===================================================================*/
/* DEP Execute Command Structures */
/*===================================================================*/

typedef struct _wfs_dep_envelope
{
    LPSTR           lpszPrintData;
    LPWSTR          lpszUNICODEPrintData;
} WFSDEPENVELOPE, *LPWFSDEPENVELOPE;

typedef struct _wfs_dep_set_guidlight
{
    WORD            wGuidLight;
    DWORD           dwCommand;
} WFSDEPSETGUIDLIGHT, *LPWFSDEPSETGUIDLIGHT;

typedef struct _wfs_dep_supply_replen
{
    WORD            fwSupplyReplen;
} WFSDEPSUPPLYREPLEN, *LPWFSDEPSUPPLYREPLEN;

typedef struct _wfs_dep_power_save_control
{
    USHORT          usMaxPowerSaveRecoveryTime;
} WFSDEPPOWERSAVECONTROL, *LPWFSDEPPOWERSAVECONTROL;

typedef struct _wfs_dep_synchronize_command
{
    DWORD           dwCommand;
    LPVOID          lpCmdData;
} WFSDEPSYNCHRONIZECOMMAND, *LPWFSDEPSYNCHRONIZECOMMAND;


/*===================================================================*/
/* DEP Message Structures */
/*===================================================================*/

typedef struct _wfs_dep_media_detected
{
    WORD            wDispenseMedia;
    WORD            wDepositMedia;
} WFSDEPMEDIADETECTED, *LPWFSDEPMEDIADETECTED;

typedef struct _wfs_dep_device_position
{
    WORD            wPosition;
} WFSDEPDEVICEPOSITION, *LPWFSDEPDEVICEPOSITION;

typedef struct _wfs_dep_power_save_change
{
    USHORT          usPowerSaveRecoveryTime;
} WFSDEPPOWERSAVECHANGE, *LPWFSDEPPOWERSAVECHANGE;


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

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

#endif /* __INC_XFSDEP__H */
```