CEN

WORKSHOP

AGREEMENT

CWA 13449-6

December 1998

ICS 35.200;35.240.15;35.240.40

English version

Extensions for Financial Services (XFS) interface specification - Part 6: PIN Keypad Device Class Interface - Programmer's Interface

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 Central Secretariat 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, Czech Republic, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

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

Central Secretariat: rue de Stassart, 36 B-1050 Brussels

Ref. No. CWA 13449-6:1998 E

Contents

Foreword

This CWA is revision 2.0 of the XFS interface specification. Release 2.0 extends the scope of the XFS interface specification to include both the self service/ATM environment as well as the branch environment. The new specification now fully supports cameras, deposit units, identification cards, PIN pads, sensors and indicator units, text terminals, cash dispenser modules and a wide variety of printing mechanisms.

This specification was originally developed by the Banking Solutions Vendor Council (BSVC), and is endorsed by the CEN/ISSS Workshop on XFS. This Workshop gathers both suppliers (among others the BSVC members) 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.

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 2.00.

This CWA is supplemented by a set of release notes, which are available from the CEN/ISSS Secretariat (an on-line version of these release notes is available from http://www.cenorm.be/isss/Workshop/XFS/release-notes.htm).

0. Introduction

This is part 6 of the multi-part CWA 13449, describing Release 2.0 of the XFS interface specification.

The full CWA 13449 "Extensions for Financial Services (XFS) interface specification"consists of the following parts:

Part 1: Application Programming Interface (API) - Service Provider Interface (SPI); Programmer's Reference
Part 2: Service Classes Definition; Programmer's Reference
Part 3: Printer 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

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 from the CEN/ISSS Secretariat (contact isss@cenorm.be or download from http://www.cenorm.be/isss/ Workshop/XFS/release-notes.htm).

The information in this document originally contributed by members of the Banking Solutions Vendor Council and endorsed by the CEN/ISSS Workshop on XFS, represents the Workshop's current views on the issues discussed as of the date of publication. It is furnished for informational purposes only and is subject to change without notice. CEN/ISSS makes no warranty, express or implied, with respect to this document.

The XFS specifications are now further developed in the CEN/ISSS Workshop on XFS. CEN/ISSS Workshops are open to all interested parties offering to contribute. Parties interested in participating should contact the CEN/ISSS Secretariat (isss@cenorm.be).

A Software Development Kit (SDK) which supplies the components and tools to allow the implementation of compliant applications and services is available from Microsoft[1].

To the extent that date processing occurs, all XFS Workshop participants agree that the XFS specifications are Year 2000 compliant.

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, with updates**
2.00	**November 11, 1996**	**Updated release encompassing self-service environment.**
	October 6, 1998	**WOSA/XFS Release 2.00 as originally developed by the BSVC, has been formally accepted as a CEN Workshop Agreement by the CEN/ISSS XFS Workshop and the name WOSA/XFS has been changed into XFS. In spite of the name change, certain occurrencies of WOSA/XFS however still appear in the documentation, for compatibility reasons**

[1] Microsoft is a registered trademark, and Windows and Windows NT are trademarks of Microsoft Corporation

1. 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 the Extensions for Financial Services specification is to standardize command 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 the union 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. Personal Identification Number (PIN) Keypads

This section describes the application program interface for personal identification number keypads (PIN pads) and other encryption/decryption devices. This description includes definitions of the service-specific commands that can be issued, using the **WFSAsyncExecute**, **WFSExecute**, **WFSGetInfo** and **WFSAsyncGetInfo** functions.

This section describes the general interface for the following functions:

- Administration of encryption devices
- Loading of encryption keys
- Encryption / decryption
- Entering Personal Identification Numbers (PINs)
- PIN verification
- PIN block generation (encrypted PIN)
- Clear text data handling
- Function key handling
- PIN presentation to chip card

If the PIN Pad device has local display capability, display handling should be handled using the Text Terminal Unit (TTU) interface.

This specification does not claim to adhere to any security standards, any security standards supported will be vendor dependent.

Important Notes:

- This revision of this specification does not define key management procedures; key management is vendor-specific.
- Key space management is customer-specific, and is therefore handled by vendor-specific mechanisms.
- Only numeric PIN pads are handled in this specification.

3. Info Commands

3.1 WFS_INF_PIN_STATUS

Description The WFS_INF_PIN_STATUS command returns several kinds of status information.

Input Param None.

Output Param

```
LPWFSPINSTATUS      lpStatus;

typedef struct _wfs_pin_status
   {
   WORD             fwDevice;
   WORD             fwEncStat;
   LPSTR            lpszExtra;
   } WFSPINSTATUS, * LPWFSPINSTATUS;
```

fwDevice
Specifies the state of the PIN pad device as one of the following flags:

Value	Meaning
WFS_PIN_DEVONLINE	The device is on-line.
WFS_PIN_DEVOFFLINE	The device is off-line.
WFS_PIN_DEVPOWEROFF	The device is powered off.
WFS_PIN_DEVBUSY	The device is busy processing a request.
WFS_PIN_DEVNODEVICE	There is no device connected.
WFS_PIN_DEVHWERROR	The device is inoperable due to a hardware error.
WFS_PIN_DEVUSERERROR	The device is inoperable due to interference by a user.

fwEncStat
Specifies the state of the Encryption Module as one of the following flags:

Value	Meaning
WFS_PIN_ENCNOTREADY	The encryption module is not ready.
WFS_PIN_ENCNOTINITIALIZED	The encryption module is not initialized (no master key loaded).
WFS_PIN_ENCINITIALIZED	The encryption module is initialized and master key (where required) and any other initial keys are loaded; ready to import other keys.
WFS_PIN_ENCREADY	The encryption module is initialized and ready (at least one key is imported into the encryption module).
WFS_PIN_ENCBUSY	The encryption module is busy (implies that the device is busy).
WFS_PIN_ENCUNDEFINED	The encryption module state is undefined.

lpszExtra
Specifies 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 extendable by service providers. Each string will be null-terminated, with the final string terminating with two null characters.

Error Codes There are no additional error codes 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.

3.2 WFS_INF_PIN_CAPABILITIES

Description This command is used to retrieve the capabilities of the PIN pad.

Input Param None.

Output Param

```
LPWFSPINCAPS lpCaps;

typedef struct _wfs_pin_caps
   {
   WORD          wClass;
   WORD          fwType;
   BOOL          bCompound;
   USHORT        usKeyNum;
   WORD          fwAlgorithms;
   WORD          fwPinFormats;
   WORD          fwDerivationAlgorithms;
   WORD          fwPresentationAlgorithms;
   WORD          fwDisplay;
   BOOL          bIDConnect;
   WORD          fwIDKey;
   WORD          fwValidationAlgorithms;
   LPSTR         lpszExtra;
   } WFSPINCAPS, * LPWFSPINCAPS;
```

wClass
Specifies the logical service class, value is:
WFS_SERVICE_CLASS_PIN

fwType
Specifies the type of the PIN pad security module as a combination of the following flags:

Value	Meaning
WFS_PIN_TYPEEPP	electronic PIN pad
WFS_PIN_TYPEEDM	encryption/decryption module

bCompound
Specifies whether the logical device is part of a compound physical device and is either TRUE or FALSE.

usKeyNum
Number of the keys which can be stored in the encryption/decryption module.

fwAlgorithms
Supported encryption modes; a combination of the following flags:

Value	Meaning
WFS_PIN_CRYPTDESECB	Electronic Code Book
WFS_PIN_CRYPTDESCBC	Cipher Block Chaining
WFS_PIN_CRYPTDESMAC	MAC calculation using CBC
WFS_PIN_CRYPTDESCFB	Cipher Feed Back
WFS_PIN_CRYPTRSA	RSA Encryption
WFS_PIN_CRYPTECMA	ECMA Encryption
WFS_PIN_CRYPTTRIDESECB	Triple DES with Electronic Code Book
WFS_PIN_CRYPTTRIDESCBC	Triple DES with Cipher Block Chaning
WFS_PIN_CRYPTTRIDESCFB	Triple DES with Cipher Feed Back
WFS_PIN_CRYPTTRIDESMAC	Triple DES MAC calculation using CBC

fwPinFormats
Supported PIN formats; a combination of the following flags:

Value	Meaning
WFS_PIN_FORM3624	PIN left justified, filled with padding characters, PIN length 4-16 digits
WFS_PIN_FORMANSI	PIN is preceded by 0x00 and the length of the PIN (0x04 to 0x0C), filled with padding character 0x0F to the right, PIN length 4-12 digits, XORed with PAN (Primary Account Number, minimum 12 digits without check

	number)
WFS_PIN_FORMISO0	PIN is preceded by 0x00 and the length of the PIN (0x04 to 0x0C), filled with padding character 0x0F to the right, PIN length 4-12 digits, XORed with PAN (Primary Account Number, no minimum length specified, missing digits are filled with 0x00)
WFS_PIN_FORMISO1	PIN is preceded by 0x01 and the length of the PIN (0x04 to 0x0C), padding characters are taken from a transaction field (10 digits)
WFS_PIN_FORMECI2	(similar to WFS_PIN_FORM3624), PIN only 4 digits
WFS_PIN_FORMECI3	PIN is preceded by the length (digit), PIN length 4-6 digits, padded with 0x00
WFS_PIN_FORMVISA	same as WFS_PIN_FORMECI3
WFS_PIN_FORMDIEBOLD	PIN is padded with the padding character and may be not encrypted, single encrypted or double encrypted.
WFS_PIN_FORMDIEBOLDCO	PIN is preceeded by the two-digit coordination number, padded with the padding character and may be not encrypted, single encrypted or double encrypted.

fwDerivationAlgorithms
Supported derivation algorithms; a combination of the following flags:

Value	Meaning
WFS_PIN_CHIP_ZKA	Algorithm for the derivation of a chip card individual key as described by the German ZKA.

fwPresentationAlgorithms
Supported presentation algorithms; a combination of the following flags:

Value	Meaning
WFS_PIN_PRESENT_CLEAR	Algorithm for the presentation of a clear text PIN to a chip card.

fwDisplay
Specifies the type of the display used in the PIN pad module as one of the following flags:

Value	Meaning
WFS_PIN_DISPNONE	no display unit
WFS_PIN_DISPLEDTHROUGH	lights next to text guide user
WFS_PIN_DISPDISPLAY	a real display is available (this doesn't apply for self-service)

bIDConnect
Specifies whether the PIN pad is directly physically connected to the ID card unit. The value of this parameter is either TRUE or FALSE.

fwIDKey
Specifies whether an ID key is supported as a combination of the following flags:

Value	Meaning
WFS_PIN_IDKEYINITIALIZATION	ID key supported in the WFS_CMD_PIN_INITIALIZATION command.
WFS_PIN_IDKEYIMPORT	ID key supported in the WFS_CMD_PIN_IMPORT_KEY command.

fwValidationAlgorithms
Specifies the algorithms for PIN validation supported by the service; combination of the following flags:

Value	Meaning
WFS_PIN_DES	DES algorithm
WFS_PIN_EUROCHEQUE	EUROCHEQUE algorithm
WFS_PIN_VISA	VISA algorithm
WFS_PIN_DES_OFFSET	DES offset generation algorithm

lpszExtra
Points 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 extendable by service providers.

Each string is null-terminated, with the final string terminating with two null characters.

Error Codes There are no additional error codes generated by this command.

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

3.3 WFS_INF_PIN_KEY_DETAIL

Description This command returns detailed information about the keys in the encryption module.

Input Param

```
LPSTR  lpsKeyName;
```

lpsKeyName
Name of the key for which detailed information is requested.
If NULL, detailed information about all the keys in the encryption module is returned.

Output Param

```
LPWFSPINKEYDETAIL * lppKeyDetail;
```

Pointer to a null-terminated array of pointers to key detail structures.

```
typedef struct _wfs_pin_key_detail
   {
   LPSTR         lpsKeyName;
   WORD          fwUse;
   BOOL          bLoaded;
   } WFSPINKEYDETAIL, * LPWFSPINKEYDETAIL;
```

lpsKeyName
Specifies the name of the key.

fwUse
Specifies the type of access for which the key is used as a combination of the following flags:

Value	Meaning
WFS_PIN_USECRYPT	key can be used for encryption/decryption
WFS_PIN_USEFUNCTION	key can be used for PIN functions
WFS_PIN_USEMACING	key can be used for MACing
WFS_PIN_USEKEYENCKEY	key is used as key encryption key
WFS_PIN_USESVENCKEY	key is used as CBC Start Value encryption key
WFS_PIN_USENODUPLICATE	key can be imported only once

bLoaded
Specifies whether the key has been loaded (imported from Application or locally from Operator) and is either TRUE or FALSE.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_KEYNOTFOUND	The specified key name is not found.

Comments None.

3.4 WFS_INF_PIN_FUNCKEY_DETAIL

Description This command returns information about the names of the Function Keys supported by the device. Location information is also returned for the supported FDKs (Function Descriptor Keys) or Touch Screen Pads if this XFS interface is used for Touch Screen input.

Input Param

```
LPULONG      lpulFDKMask;
```

lpulFDKMask
Mask for the FDKs for which additional information is requested.
If 0x00000000, only information about function keys is returned.
If 0xFFFFFFFF, information about all the supported FDKs is returned.

Output Param

```
LPWFSPINFUNCKEYDETAIL      lpFuncKeyDetail;

typedef struct _wfs_pin_func_key_detail
   {
   ULONG            ulFuncMask;
   USHORT           usNumberFDKs;
   LPWFSPINFDK      * lppFDKs;
   } WFSPINFUNCKEYDETAIL, * LPWFSPINFUNCKEYDETAIL;
```

ulFuncMask
Specifies the function keys available for this physical device as a combination of the following flags:

WFS_PIN_FK_0	
WFS_PIN_FK_1	
WFS_PIN_FK_2	
WFS_PIN_FK_3	
WFS_PIN_FK_4	
WFS_PIN_FK_5	
WFS_PIN_FK_6	
WFS_PIN_FK_7	
WFS_PIN_FK_8	
WFS_PIN_FK_9	
WFS_PIN_FK_ENTER	
WFS_PIN_FK_CANCEL	
WFS_PIN_FK_CLEAR	
WFS_PIN_FK_BACKSPACE	
WFS_PIN_FK_HELP	
WFS_PIN_FK_DECPOINT	
WFS_PIN_FK_00	
WFS_PIN_FK_000	
WFS_PIN_FK_RES1	(reserved for future use)
WFS_PIN_FK_RES2	(reserved for future use)
WFS_PIN_FK_RES3	(reserved for future use)
WFS_PIN_FK_RES4	(reserved for future use)
WFS_PIN_FK_RES5	(reserved for future use)
WFS_PIN_FK_RES6	(reserved for future use)
WFS_PIN_FK_RES7	(reserved for future use)
WFS_PIN_FK_RES8	(reserved for future use)

The remaining 6 bit masks may be used as vendor dependent keys.

WFS_PIN_FK_OEM1	
WFS_PIN_FK_OEM2	
WFS_PIN_FK_OEM3	
WFS_PIN_FK_OEM4	
WFS_PIN_FK_OEM5	
WFS_PIN_FK_OEM6	

usNumberFDKs
This value indicates the number of FDK structures returned. This number can be less than the number of keys requested, if any keys are not supported.

lppFDKs
Pointer to an array of pointers to FDK structures.

```
typedef struct _wfs_pin_fdk
   {
   ULONG     ulFDK;
   USHORT    usXPosition;
   USHORT    usYPosition;
   } WFSPINFDK, * LPWFSPINFDK;
```

ulFDK
Specifies the code returned by this FDK, defined as one of the following values:

WFS_PIN_FK_FDK01
WFS_PIN_FK_FDK02
WFS_PIN_FK_FDK03
WFS_PIN_FK_FDK04
WFS_PIN_FK_FDK05
WFS_PIN_FK_FDK06
WFS_PIN_FK_FDK07
WFS_PIN_FK_FDK08
WFS_PIN_FK_FDK09
WFS_PIN_FK_FDK10
WFS_PIN_FK_FDK11
WFS_PIN_FK_FDK12
WFS_PIN_FK_FDK13
WFS_PIN_FK_FDK14
WFS_PIN_FK_FDK15
WFS_PIN_FK_FDK16
WFS_PIN_FK_FDK17
WFS_PIN_FK_FDK18
WFS_PIN_FK_FDK19
WFS_PIN_FK_FDK20
WFS_PIN_FK_FDK21
WFS_PIN_FK_FDK22
WFS_PIN_FK_FDK23
WFS_PIN_FK_FDK24
WFS_PIN_FK_FDK25
WFS_PIN_FK_FDK26
WFS_PIN_FK_FDK27
WFS_PIN_FK_FDK28
WFS_PIN_FK_FDK29
WFS_PIN_FK_FDK30
WFS_PIN_FK_FDK31
WFS_PIN_FK_FDK32

usXPosition
For FDKs, specifies the FDK position relative to the Left Hand side of the screen expressed as a percentage of the width of the screen.

usYPosition
For FDKs, specifies the FDK position relative to the top of the screen expressed as a percentage of the height of the screen.

Error Codes There are no additonal error codes generated by this command.

Comments None.

4. Execute Commands

4.1 WFS_CMD_PIN_CRYPT

Description The input data is either encrypted or decrypted using the specified or selected encryption mode. The available modes are defined in the WFS_INF_PIN_CAPABILITIES command.

This command can also be used for Message Authentication Code generation (i.e. MACing). For this purpose, it is possible to specify how the data is formatted before the encryption.

The input data can be expanded with a fill-character to the necessary length (mandated by the encryption algorithm being used).

The Start Value (or Initialization Vector) should be able to be passed encrypted like the specified encryption/decryption key. It would therefore need to be decrypted with a loaded key so the name of this key must also be passed. However, both these parameters are optional.

Input Param `LPWFSPINCRYPT lpCrypt;`

```
typedef struct _wfs_pin_crypt
   {
   WORD             wMode;
   LPSTR            lpsKey;
   LPWFSXDATA       lpxKeyEncKey;
   WORD             wAlgorithm;
   LPSTR            lpsStartValueKey;
   LPWFSXDATA       lpxStartValue;
   BYTE             bPadding;
   BYTE             bCompression;
   LPWFSXDATA       lpxCryptData;
   } WFSPINCRYPT, * LPWFSPINCRYPT;
```

wMode
Specifies whether to encrypt or decrypt, values are one of the following:

Value	Meaning
WFS_PIN_MODEENCRYPT	encrypt with key
WFS_PIN_MODEDECRYPT	decrypt with key

This parameter does not apply to MACing.

lpsKey
Specifies the name of the stored key.

lpxKeyEncKey
If NULL, *lpsKey* is used directly for encryption/decryption. Otherwise, *lpsKey* is used to decrypt the encrypted key passed in *lpxKeyEncKey* and the result is used for encryption/decryption. Key is a double length key when used for Triple DES encryption/decryption. Users of this specification must adhere to local regulations when using Triple DES.

wAlgorithm
Specifies the encryption algorithm. Possible values are those described in WFS_INF_PIN_CAPABILITIES.

lpsStartValueKey
Specifies the name of the stored key used to decrypt the *lpxStartValue* to obtain the Initialization Vector. If this parameter is NULL, *lpxStartValue* is used as the Initialization Vector.

lpxStartValue
DES and Triple DES initialization vector for CBC / CFB encryption and MACing. If this parameter is NULL *lpsStartValueKey* is used as the Start Value. If *lpsStartValueKey* is also NULL, the default value for CBC / CFB / MAC is 16 hex digits 0x0.

bPadding
Specifies the padding character for encryption.

bCompression
Specifies whether data is to be compressed (blanks removed) before building the MAC. If *bCompression* is 0x00 no compression is selected, otherwise *bCompression* holds the representation of the blank character in the actual code table.

lpxCryptData
Pointer to the data to be encrypted, decrypted, or MACed.

Output Param `LPWFSXDATA lpxCryptData;`

lpxCryptData
Pointer to the encrypted or decrypted data, or MAC value.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_KEYNOTFOUND	The specified key was not found.
WFS_ERR_PIN_KEYNOVALUE	The specified key is not loaded.
WFS_ERR_PIN_USEVIOLATION	The specified use is not supported by this key.
WFS_ERR_PIN_MODENOTSUPPORTED	The specified mode is not supported.
WFS_ERR_PIN_ACCESSDENIED	The encryption module is either not initialized or not ready for any vendor specific reason.
WFS_ERR_PIN_INVALIDKEYLENGTH	The length of *lpxKeyEncKey* or *lpxStartValue* is not supported.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_SRVE_PIN_ILLEGAL_KEY_ACCESS	An error occured accessing an encryption key.

Comments The datatype LPWFSXDATA is used to pass hexadecimal data and is defined as follows :

```
typedef struct _wfs_hex_data
   {
   USHORT    usLength;
   LPBYTE    lpbData;
   } WFSXDATA, *LPWFSXDATA;
```

usLength
Length of the byte stream pointed to by *lpbData.*

lpbData
Pointer to the binary data stream.

4.2 WFS_CMD_PIN_IMPORT_KEY

Description The key passed by the application is loaded in the encryption module. The key can be passed in clear text mode or encrypted with an accompanying “key encryption key”.

Input Param
```
LPWFSPINIMPORT      lpImport;

typedef struct _wfs_pin_import
   {
   LPSTR            lpsKey;
   LPSTR            lpsEncKey;
   LPWFSXDATA       lpxIdent;
   LPWFSXDATA       lpxValue;
   WORD             fwUse;
   } WFSPINIMPORT, * LPWFSPINIMPORT;
```

lpsKey
Specifies the name of key being loaded.

lpsEncKey
If *lpsEncKey* is NULL the key is loaded directly into the encryption module. Otherwise, *lpsEncKey* specifies a key name or a format name which were used to encrypt the key string passed in *lpxValue*.

lpxIdent
Specifies the key owner identification. The use of this parameter is vendor dependent.

lpxValue
Specifies the value of key to be loaded.

fwUse
Specifies the type of access for which the key can be used as a combination of the following flags:

Value	Meaning
WFS_PIN_USECRYPT	key can be used for encryption/decryption
WFS_PIN_USEFUNCTION	key can be used for PIN functions
WFS_PIN_USEMACING	key can be used for MACing
WFS_PIN_USEKEYENCKEY	key is used as key encryption key
WFS_PIN_USESVENCKEY	key is used as CBC Start Value encryption key
WFS_PIN_USENODUPLICATE	key can be imported only once

Output Param `LPWFSXDATA lpxKVC;`

lpxKVC
pointer to the key verification code data that can be used for verification of the loaded key, NULL if device does not have that capability.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_KEYNOTFOUND	The specified key encryption key was not found.
WFS_ERR_PIN_KEYNOVALUE	The specified key encryption key is not loaded.
WFS_ERR_PIN_USEVIOLATION	The specified use is not supported by this key.
WFS_ERR_PIN_ACCESSDENIED	The encryption module is either not initialized or not ready for any vendor specific reason.
WFS_ERR_PIN_INVALIDID	The ID passed was not valid.
WFS_ERR_PIN_DUPLICATEKEY	A key exists with that name and cannot be overwritten.
WFS_ERR_PIN_INVALIDKEYLENGTH	The length of *lpxValue* is not supported.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_SRVE_PIN_ILLEGAL_KEY_ACCESS	An error occured accessing an encryption key.

Comments None.

4.3 WFS_CMD_PIN_DERIVE_KEY

Description A key is derived from input data using a key generating key and an initialization vector. The input data can be expanded with a fill-character to the necessary length (mandated by the encryption algorithm being used). The derived key is imported into the encryption module and is used for encryption or decryption operations.

Input Param `LPWFSPINDERIVE lpDerive;`

```
typedef struct _wfs_pin_derive
   {
   WORD            wDerivationAlgorithm;
   LPSTR           lpsKey;
   LPSTR           lpsKeyGenKey;
   LPSTR           lpsStartValueKey;
   LPWFSXDATA      lpxStartValue;
   BYTE            bPadding;
   LPWFSXDATA      lpxInputData;
   LPWFSXDATA      lpxIdent;
   } WFSPINDERIVE, * LPWFSPINDERIVE;
```

wDerivationAlgorithm
Specifies the algorithm that is used for derivation. Possible values are:
(see command WFS_INF_PIN_CAPABILITIES)

lpsKey
Specifies the name where the derived key will be stored.

lpsKeyGenKey
Specifies the name of the key generating key that is used for the derivation.

lpsStartValueKey
Specifies the name of the stored key used to decrypt the *lpxStartValue* to obtain the Initialization Vector. If this parameter is NULL, *lpxStartValue* is used as the Initialization Vector.

lpxStartValue
DES initialization vector for the encryption step within the derivation.

bPadding
Specifies the padding character for the encryption step within the derivation.

lpxInputData
Pointer to the data to be used for key derivation.

lpxIdent
Specifies the key owner identification. The use of this parameter is vendor dependent.

Output Param None.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_KEYNOTFOUND	The specified key was not found.
WFS_ERR_PIN_KEYNOVALUE	The specified key is not loaded.
WFS_ERR_PIN_USEVIOLATION	The specified use is not supported by this key.
WFS_ERR_PIN_MODENOTSUPPORTED	The specified algorithm is not supported.
WFS_ERR_PIN_ACCESSDENIED	The encryption module is either not initialized (or not ready for some vendor specific reason).
WFS_ERR_PIN_INVALIDID	The ID passed was not valid.
WFS_ERR_PIN_DUPLICATEKEY	A key exists with that name and cannot be overwritten.
WFS_ERR_PIN_INVALIDKEYLENGTH	The length of *lpxStartValue* is not supported.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_SRVE_PIN_ILLEGAL_KEY_ACCESS	An error occured accessing an encryption key.

Comments None.

4.4 WFS_CMD_PIN_GET_PIN

Description This function stores the PIN entry via the PIN pad. From the point this function is invoked, PIN digit entries are ***not*** passed to the application. For each PIN digit, or any other active key entered, an execute notification event is sent in order to allow an application to perform the appropriate display action (i.e. when the PIN pad has no integrated display). The application is not informed of the value entered, the execute notification only informs that a key has been depressed.

Some PIN pad devices do not inform the application as each PIN digit is entered, but locally process the PIN entry based upon minimum PIN length and maximum PIN length input parameters. These PIN pad devices which provide local PIN entry management and optional display tracking may or may not notify the application of a minimum PIN length violation.

When the maximum number of PIN digits is entered, or a completion key is pressed after the minimum number of PIN digits is entered, a WFS_EXEC_COMPLETE event message is sent to

the application. Once this notification is received, the output parameters are then returned to the application from this function call. The depression of the <Cancel> key is also passed to the application via the WFS_EXEC_COMPLETE event message.

If *usMaxLen* is zero, the service provider does not terminate the command unless the application sets *ulTerminateKeys* or *ulTerminateFDKs*. In the event that *ulTerminateKeys* or *ulTerminateFDKs* are not set and *usMaxLen* is zero, the command will not terminate and the application must issue a WFSCancel command.

Input Param

```
LPWFSPINGETPIN      lpGetPin;

typedef struct _wfs_pin_getpin
   {
   USHORT    usMinLen;
   USHORT    usMaxLen;
   BOOL      bAutoEnd;
   CHAR      cEcho;
   ULONG     ulActiveFDKs;
   ULONG     ulActiveKeys;
   ULONG     ulTerminateFDKs;
   ULONG     ulTerminateKeys;
   } WFSPINGETPIN, * LPWFSPINGETPIN;
```

usMinLen
Specifies the minimum number of digits which must be entered for the PIN. A value of zero indicates no minimum PIN length verification.

usMaxLen
Specifies the maximum number of digits which can be entered for the PIN.

bAutoEnd
If *bAutoEnd* is set to true, the service provider terminates the command when the maximum number of digits are entered. Otherwise, the input is terminated by the user using one of the termination keys. When *usMaxLen* is reached, the service provider will disable all numeric keys. *bAutoEnd* is ignored when *usMaxLen* is set to 0.

cEcho
Specifies the replace character to be echoed on a local display for the PIN digit.

ulActiveFDKs
Specifies those FDKs which are active during the execution of the command.

ulActiveKeys
Specifies those (other) Function Keys which are active during the execution of the command.

ulTerminateFDKs
Specifies those FDKs which must terminate the execution of the command.

ulTerminateKeys
Specifies those (other) Function Keys which must terminate the execution of the command.

Output Param

```
LPWFSPINENTRY lpEntry;

typedef struct _wfs_pin_entry
   {
   USHORT    usDigits;
   WORD      wCompletion;
   } WFSPINENTRY, * LPWFSPINENTRY;
```

usDigits
Specifies the number of PIN digits entered.

wCompletion
Specifies the reason for completion of the entry. Possible values are:

Value	Meaning
WFS_PIN_COMPAUTO	The command terminated automatically, because maximum PIN length was reached.
WFS_PIN_COMPENTER	The ENTER Function Key was pressed.

WFS_PIN_COMPCANCEL	The CANCEL Function Key was pressed.
WFS_PIN_COMPCONTINUE	Input continues (this value is only used in the execute event WFS_EXEE_PIN_KEY).
WFS_PIN_COMPCLEAR	The CLEAR Function Key was pressed and the previous input is cleared (this value is only used in the execute event WFS_EXEE_PIN_KEY).
WFS_PIN_COMPBACKSPACE	The last input digit was cleared (this value is only used in the execute event WFS_EXEE_PIN_KEY).
WFS_PIN_COMPFDK	An FDK was pressed.
WFS_PIN_COMPHELP	The HELP Function Key was pressed..
WFS_PIN_COMPFK	A Function Key (FK) other than ENTER, CLEAR, CANCEL, BACKSPACE, HELP was pressed.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_KEYINVALID	At least one of the specified function keys or FDKs is invalid.
WFS_ERR_PIN_KEYNOTSUPPORTED	At least one of the specified function keys or FDKs is not supported by the service provider.
WFS_ERR_PIN_NOACTIVEKEYS	There are no active function keys specified.
WFS_ERR_PIN_NOTERMINATEKEYS	There are no terminate keys specified and usMaxLen is set to 0.
WFS_ERR_PIN_MINIMUMLENGTH	The minimum PIN length field is invalid or greater than the maximum PIN length field.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_EXEE_PIN_KEY	A key has been pressed at the PIN pad.

Comments None.

4.5 WFS_CMD_PIN_LOCAL_DES

Description The PIN, which was entered with the WFS_PIN_GET_PIN command, is combined with the requisite data specified by the DES validation algorithm and locally verified for correctness. The local DES verification is based on the IBM 3624 standard. The result of the verification is returned to the application. This command will clear the PIN.

Input Param

```
LPWFSPINLOCALDES    lpLocalDES;

typedef struct _wfs_pin_local_des
   {
   LPSTR         lpsValidationData;
   LPSTR         lpsOffset;
   BYTE          bPadding;
   USHORT        usMaxPIN;
   USHORT        usValDigits;
   BOOL          bNoLeadingZero;
   LPSTR         lpsKey;
   LPWFSXDATA    lpxKeyEncKey;
   LPSTR         lpsDecTable;
   } WFSPINLOCALDES, * LPWFSPINLOCALDES;
```

lpsValidationData
Validation data

lpsOffset
Offset for the PIN block; if NULL then no offset is used.

bPadding
Specifies the padding character for validation data.

usMaxPIN
Maximum number of PIN digits to be used for validation.

usValDigits
Number of Validation digits to be used for validation.

bNoLeadingZero
If set to TRUE and the first digit of result of the modulo 10 addition is a X'0', it is replaced with X'1' before performing the verification against the entered PIN. If set to FALSE, a leading zero is allowed in entered PINs.

lpsKey
Name of the validation key

lpxKeyEncKey
If NULL, *lpsKey* is used directly for PIN validation. Otherwise, *lpsKey* is used to decrypt the encrypted key passed in *lpxKeyEncKey* and the result is used for PIN validation.

lpsDecTable
ASCII decimalization table (16 character string containing characters '0' to '9'). Used to convert the hexadecimal digits (0x0 to 0xF) of the encrypted validation data to decimal digits (0x0 to 0x9).

Output Param `LPBOOL pbResult;`

lpbResult
Pointer to a boolean value which specifies whether the PIN is correct or not.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_KEYNOTFOUND	The specified key was not found.
WFS_ERR_PIN_KEYNOVALUE	The specified key is not loaded.
WFS_ERR_PIN_USEVIOLATION	The specified use is not supported by this key.
WFS_ERR_PIN_ACCESSDENIED	The encryption module is either not initialized or not ready for any vendor specific reason.
WFS_ERR_PIN_NOPIN	PIN has not been entered or has been cleared.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_SRVE_PIN_ILLEGAL_KEY_ACCESS	An error occured accessing an encryption key.

Comments None.

4.6 WFS_CMD_PIN_CREATE_OFFSET

Description This function is used to generate a PIN Offset that is used to verify PINs using the WFS_CMD_PIN_LOCAL_DES execute command. The PIN offset is computed by combining validation data with the keypad entered PIN. This command will clear the PIN

Input Param `LPWFSPINCREATEOFFSET lpPINOffset;`

```
typedef struct _wfs_pin_create_offset
   {
   LPSTR         lpsValidationData;
   BYTE          bPadding;
   USHORT        usMaxPIN;
   USHORT        usValDigits;
   LPSTR         lpsKey;
   LPWFSXDATA    lpxKeyEncKey;
   LPSTR         lpsDecTable;
   } WFSPINCREATEOFFSET, * LPWFSPINCREATEOFFSET;
```

lpsValidationData
Validation data

bPadding
Specifies the padding character for validation data.

usMaxPIN
Maximum number of PIN digits to be used for PIN Offset creation.

usValDigits
Number of Validation Data digits to be used for PIN Offset creation.

lpsKey
Name of the validation key

lpxKeyEncKey
If NULL, *lpsKey* is used directly in PIN Offset creation. Otherwise, *lpsKey* is used to decrypt the encrypted key passed in *lpxKeyEncKey* and the result is used in PIN Offset creation.

lpsDecTable
ASCII decimalization table (16 character string containing characters '0' to '9'). Used to convert the hexadecimal digits (0x0 to 0xF) of the encrypted validation data to decimal digits (0x0 to 0x9).

Output Param `LPSTR lpsOffset;`

lpsOffset
Computed PIN Offset.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_KEYNOTFOUND	The specified key was not found.
WFS_ERR_PIN_KEYNOVALUE	The specified key is not loaded.
WFS_ERR_PIN_USEVIOLATION	The specified use is not supported by this key.
WFS_ERR_PIN_ACCESSDENIED	The encryption module is either not initialized or not ready for any vendor specific reason.
WFS_ERR_PIN_NOPIN	PIN has not been entered or has been cleared.
WFS_ERR_PIN_NOTALLOWED	PIN entered by the user is not allowed.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_SRVE_PIN_ILLEGAL_KEY_ACCESS	An error occured accessing an encryption key.

Comments The list of 'forbidden' PINs (values that cannot be chosen as a PIN, e.g. 1111) is configured in the device in a vendor dependent way during the configuration of the system.

4.7 WFS_CMD_PIN_LOCAL_EUROCHEQUE

Description The PIN, which was entered with the WFS_PIN_GET_PIN command, is combined with the requisite data specified by the Eurocheque validation algorithm and locally verified for correctness. The result of the verification is returned to the application. This command will clear the PIN.

Input Param `LPWFSPINLOCALEUROCHEQUE lpLocalEurocheque;`

```
typedef struct _wfs_pin_local_eurocheque
   {
   LPSTR         lpsEurochequeData;
   LPSTR         lpsPVV;
   WORD          wFirstEncDigits;
   WORD          wFirstEncOffset;
   WORD          wPVVDigits;
   WORD          wPVVOffset;
   LPSTR         lpsKey;
   LPWFSXDATA    lpxKeyEncKey;
   LPSTR         lpsDecTable;
   } WFSPINLOCALEUROCHEQUE, * LPWFSPINLOCALEUROCHEQUE;
```

lpsEurochequeData
Track-3 Eurocheque data

lpsPVV
PIN Validation Value from track data.

wFirstEncDigits
Number of digits to extract after first encryption.

wFirstEncOffset
Offset of digits to extract after first encryption.

wPVVDigits
Number of digits to extract for PVV.

wPVVOffset
Offset of digits to extract for PVV.

lpsKey
Name of the validation key.

lpxKeyEncKey
If NULL, *lpsKey* is used directly for PIN validation. Otherwise, *lpsKey* is used to decrypt the encrypted key passed in *lpxKeyEncKey* and the result is used for PIN validation.

lpsDecTable
ASCII decimalization table (16 character string containing characters '0' to '9'). Used to convert the hexadecimal digits (0x0 to 0xF) of the encrypted validation data to decimal digits (0x0 to 0x9).

Output Param `LPBOOL lpbResult;`

lpbResult
Pointer to a boolean value which specifies whether the PIN is correct or not.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_KEYNOTFOUND	The specified key was not found.
WFS_ERR_PIN_KEYNOVALUE	The specified key is not loaded.
WFS_ERR_PIN_USEVIOLATION	The specified use is not supported by this key.
WFS_ERR_PIN_ACCESSDENIED	The encryption module is either not initialized or not ready for any vendor specific reason.
WFS_ERR_PIN_NOPIN	PIN has not been entered or has been cleared.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_SRVE_PIN_ILLEGAL_KEY_ACCESS	An error occured accessing an encryption key.

Comments None.

4.8 WFS_CMD_PIN_LOCAL_VISA

Description The PIN, which was entered with the WFS_PIN_GET_PIN command, is combined with the requisite data specified by the VISA validation algorithm and locally verified for correctness. The result of the verification is returned to the application. This command will clear the PIN.

Input Param `LPWFSPINLOCALVISA lpLocalVISA;`

```
typedef struct _wfs_pin_local_visa
   {
   LPSTR         lpsPAN;
   LPSTR         lpsPVV;
   WORD          wPVVDigits;
   LPSTR         lpsKey;
   LPWFSXDATA    lpxKeyEncKey;
   } WFSPINLOCALVISA, * LPWFSPINLOCALVISA;
```

lpsPAN
Primary Account Number from track data.

lpsPVV
PIN Validation Value from track data.

wPVVDigits
Number of digits of PVV.

lpsKey
Name of the validation key.

lpxKeyEncKey
If NULL, *lpsKey* is used directly for PIN validation. Otherwise, *lpsKey* is used to decrypt the encrypted key passed in *lpxKeyEncKey* and the result is used for PIN validation.

Output Param
```
LPBOOL        lpbResult;
```

lpbResult
Pointer to a boolean value which specifies whether the PIN is correct or not.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_KEYNOTFOUND	The specified key was not found.
WFS_ERR_PIN_KEYNOVALUE	The specified key is not loaded.
WFS_ERR_PIN_USEVIOLATION	The specified use is not supported by this key.
WFS_ERR_PIN_ACCESSDENIED	The encryption module is either not initialized or not ready for any vendor specific reason.
WFS_ERR_PIN_NOPIN	PIN has not been entered or has been cleared.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_SRVE_PIN_ILLEGAL_KEY_ACCESS	An error occured accessing an encryption key.

Comments None.

4.9 WFS_CMD_PIN_PRESENT_IDC

Description The PIN, which was entered with the WFS_PIN_GET_PIN command, is combined with the requisite data specified by the IDC presentation algorithm and presented to the smartcard contained in the ID Card unit. The result of the presentation is returned to the application. This command will clear the PIN.

Input Param
```
LPWFSPINPRESENTIDC  lpPresentIDC;

typedef struct _wfs_pin_presentidc
   {
   WORD          wPresentAlgorithm;
   WORD          wChipProtocol;
   ULONG         ulChipDataLength;
   LPBYTE        lpbChipData;
   LPVOID        lpAlgorithmData;
   } WFSPINPRESENTIDC, * LPWFSPINPRESENTIDC;
```

wPresentAlgorithm
Specifies the algorithm that is used for presentation. Possible values are: (see command WFS_INF_PIN_CAPABILITIES).

wChipProtocol
Identifies the protocol that is used to communicate with the chip. Possible values are: (see command WFS_INF_IDC_CAPABILITIES in the Identification Card Device Class Interface).

ulChipDataLength
Specifies the length of the byte stream pointed to by *lpbChipData*.

lpbChipData
Points to the data to be sent to the chip.

lpAlgorithmData
Pointer to a structure that contains the data required for the specified presentation algorithm. For the WFS_PIN_PRESENT_CLEAR algorithm, this structure is defined as:

```
typedef struct _wfs_pin_presentclear
  {
  ULONG         ulPINPointer;
  USHORT        usPINOffset;
  } WFSPINPRESENTCLEAR, * LPWFSPINPRESENTCLEAR;
```

ulPINPointer
Describes the byte position where to insert the PIN in the *lpbChipData* buffer. The first byte of the *lpbChipData* buffer is numbered 0.

usPINOffset
Describes the bit position where to insert the PIN in the *lpbChipData* buffer. In each byte, the most-significant bit is numbered 0, the less significant bit is numbered 7.

Output Param `LPWFSPINPRESENTRESULT lpPresentResult;`

```
typedef struct _wfs_pin_present_result
   {
   WORD      wChipProtocol;
   ULONG     ulChipDataLength;
   LPBYTE    lpbChipData;
   } WFSPINPRESENTRESULT, * LPWFSPINPRESENTRESULT;
```

wChipProtocol
Identifies the protocol that was used to communicate with the chip. This field contains the same value as the corresponding field in the input structure.

ulChipDataLength
Specifies the length of the byte stream pointed to by *lpbChipData*.

lpbChipData
Points to the data responded from the chip.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_INVALIDDATA	An error occurred while communicating with the chip.
WFS_ERR_PIN_PROTOCOLNOTSUPP	The specified protocol is not supported by the service provider.
WFS_ERR_PIN_NOPIN	PIN has not been entered or has been cleared.
WFS_ERR_PIN_ACCESSDENIED	The ID card unit is not ready for PIN presentation or for any vendor specific reason. The ID card service provider, if any, may have generated a service event that further describes the reason for that error code.

Events There are no additional events generated by this command.

Comments None.

4.10 WFS_CMD_PIN_GET_PINBLOCK

Description This function takes the account information and a PIN entered by the user to build a formatted PIN. Encrypting this formatted PIN once or twice returns a PIN block which can be written on a magnetic card or sent to a host. The PIN block can be calculated using one of the formats specified in the WFS_INF_PIN_CAPABILITIES command. This command clears the PIN.

Input Param `LPWFSPINBLOCK lpPinBlock;`

```
typedef struct _wfs_pin_block
   {
   LPSTR        lpsCustomerData;
   LPSTR        lpsXORData;
   BYTE         bPadding;
   WORD         wFormat;
   LPSTR        lpsKey;
   LPSTR        lpsKeyEncKey;
   } WFSPINBLOCK, * LPWFSPINBLOCK;
```

lpsCustomerData
Used for ANSI, ISO-0 and ISO-1 algorithm to build the formatted PIN. For ANSI and ISO-0 the PAN (Primary Account Number) is used, for ISO-1 a ten digit transaction field is required. If not used a NULL is required.
Used for DIEBOLD with coordination number, as a two digit coordination number.

lpsXORData
If the formatted PIN is encrypted twice to build the resulting PIN block, this data can be used to modify the result of the first encryption by an XOR-operation.

bPadding
Specifies the padding character.

wFormat
Specifies the format of the PIN block. Possible values are:
(see command WFS_INF_PIN_CAPABILITIES)

lpsKey
Specifies the key used to encrypt the formatted pin for the first time, NULL if no encryption is required.

lpsEncKey
Specifies the key used to format the once encrypted formatted PIN, NULL if no second encryption required.

Output Param `LPWFSXDATA  lpxPinBlock;`

lpxPinBlock
Pointer to the encrypted/decrypted data.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_KEYNOTFOUND	The specified key was not found
WFS_ERR_PIN_KEYNOVALUE	The specified key is not loaded.
WFS_ERR_PIN_USEVIOLATION	The specified use is not supported by this key.
WFS_ERR_PIN_MODENOTSUPPORTED	The specified mode is not supported.
WFS_ERR_PIN_ACCESSDENIED	The encryption module is either not initialized or not ready for any vendor specific reason.
WFS_ERR_PIN_NOPIN	PIN has been cleared.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_SRVE_PIN_ILLEGAL_KEY_ACCESS	An error occured accessing an encryption key.

Comments None.

4.11 WFS_CMD_PIN_GET_DATA

Description This function is used to return keystrokes entered by the user. It will automatically set the PIN pad to echo characters on the display if there is a display. For each keystroke an execute notification event is sent in order to allow an application to perform the appropriate display action (i.e. when the PIN pad has no integrated display).

If *usMaxLen* is zero, the service provider does not terminate the command unless the application sets *ulTerminateKeys* or *ulTerminateFDKs*. In the event that *ulTerminateKeys* or *ulTerminateFDKs* are not set and *usMaxLen* is zero, the command will not terminate and the application must issue a WFSCancel command.

Input Param

```
LPWFSPINGETDATA     lpPinGetData;

typedef struct _wfs_pin_getdata
   {
   USHORT    usMaxLen;
   BOOL      bAutoEnd;
   ULONG     ulActiveFDKs;
   ULONG     ulActiveKeys;
   ULONG     ulTerminateFDKs;
   ULONG     ulTerminateKeys;
   } WFSPINGETDATA, * LPWFSPINGETDATA;
```

usMaxLen
Specifies the maximum number of digits which can be returned to the application in the data buffer.

bAutoEnd
If *bAutoEnd* is set to true, the service provider terminates the command when the maximum number of digits are entered. Otherwise, the input is terminated by the user using one of the termination keys. When *usMaxLen* is reached, the service provider will disable all numeric keys. *bAutoEnd* is ignored when *usMaxLen* is set to 0.

ulActiveFDKs
Specifies those FDKs which are active during the execution of the command.

ulActiveKeys
Specifies those (other) Function Keys which are active during the execution of the command.

ulTerminateFDKs
Specifies those FDKs which must terminate the execution of the command.

ulTerminateKeys
Specifies those (other) Function Keys which must terminate the execution of the command.

Output Param

```
LPWFSPINDATA lpPinData;

typedef struct _wfs_pin_data
   {
   LPSTR        lpsData;
   WORD         wCompletion;
   } WFSPINDATA, * LPWFSPINDATA
```

lpsData
Pointer to the data entered by the user. This pointer is set to NULL if *usMaxLen* is set to 0.

wCompletion
Specifies the reason for completion of the entry. Possible values are:
(see command WFS_CMD_PIN_GET_PIN)

Error Codes

The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_KEYINVALID	At least one of the specified function keys or FDKs is invalid.
WFS_ERR_PIN_KEYNOTSUPPORTED	At least one of the specified function keys or FDKs is not supported by the service provider.
WFS_ERR_PIN_NOACTIVEKEYS	There are no active function keys specified.

Events

The following additional events can be generated by this command:

Value	Meaning
WFS_EXEE_PIN_KEY	A key has been pressed at the PIN pad.

Comments

None.

4.12 WFS_CMD_PIN_INITIALIZATION

Description The encryption module must be initialized before any encryption function can be used. Every initialization destroys all keys that have been loaded or imported. Usually this command is called by an operator task and not by the application program.

Initialization also involves loading "initial" application keys and local vendor dependent keys. These can be supplied, for example, by an operator through a keyboard, a local configuration file or possibly by means of some secure hardware that can be attached to the device. The application "initial" keys would normally get updated by the application during a WFS_EXEC_PIN_IMPORT command as soon as possible. Local vendor dependent static keys (e.g. storage, firmware and offset keys) would normally be transparent to the application and by definition can not be dynamically changed.

Where initial keys are not available immediately when this command is issued (i.e. when operator intervention is required), the Service Provider returns WFS_ERR_PIN_ACCESS_DENIED and the application must await the WFS_SRVE_PIN_INITIALIZED event.

During initialization an optional encrypted ID key can be stored in the HW module. The ID key and the corresponding encryption key can be passed as parameters; if not, they are generated automatically by the encryption module. The encrypted ID is returned to the application and serves as authorization for the key import function. The WFS_INF_PIN_CAPABILITIES command indicates whether or not the device will support this feature.

Input Param

```
LPWFSPININIT lpInit;

typedef struct _wfs_pin_init
   {
   LPWFSXDATA       lpxIdent;
   LPWFSXDATA       lpxKey;
   } WFSPININIT, * LPWFSPININIT;
```

lpxIdent
Pointer to the value of the ID key. Null if not required.

lpxKey
Pointer to the value of the encryption key. Null if not required.

Output Param

```
LPWFSXDATA    lpxIdentification;
```

lpxIdentification
Pointer to the value of the ID key encrypted by the encryption key. Can be used as authorization for the WFS_CMD_PIN_IMPORT_KEY command, can be NULL if no authorization required.

Error Codes The following additional error codes can be generated by this command:

Value	Meaning
WFS_ERR_PIN_ACCESSDENIED	The encryption module is either not initialized (or not ready for some vendor specific reason).
WFS_ERR_PIN_INVALIDID	The ID passed was not valid.

Events The following additional events can be generated by this command:

Value	Meaning
WFS_SRVE_PIN_ILLEGAL_KEY_ACCESS	An error occured accessing an encryption key.
WFS_SRVE_PIN_INITIALIZED	The encryption module is now initialized.

Comments None.

5. Events

5.1 WFS_EXEE_PIN_KEY

Description This event specifies that a key has been pressed at the PIN pad. It is used if the device has no internal display unit and the application has to manage the display of the entered digits.

Event Param

```
LPWFSPINKEY   lpKey;

typedef struct _wfs_pin_key
   {
   WORD      wCompletion;
   ULONG     ulDigit;
   } WFSPINKEY, * LPWFSPINKEY;
```

wCompletion
Specifies the reason for completion of the entry. Possible values are:
(see command WFS_CMD_PIN_GET_PIN)

ulDigit
Specifies the digit entered by the user or the replace character when working in encryption mode (WFS_CMD_PIN_GET_PIN). If no digit but a function key has been depressed, the key code is returned in this parameter.

Comments None.

5.2 WFS_SRVE_PIN_INITIALIZED

Description This event specifies that, as a result of a WFS_CMD_PIN_INITIALIZATION, the encryption module is now initialized and the master key (where required) and any other initial keys are loaded; ready to import other keys.

Event Param

```
LPWFSPININIT  lpInit;
```

lpInit
For a definition of WFSPININIT see command WFS_CMD_PIN_INITIALIZATION.

Comments None.

5.3 WFS_SRVE_PIN_ILLEGAL_KEY_ACCESS

Description This event specifies that an error occurred accessing an encryption key. Possible situations for generating this event are the encryption key was not found, had no value, or a use violation.

Event Param

```
LPWFSPINACCESS      lpAccess;

typedef struct _wfs_pin_access
   {
   LPSTR     lpsKeyName;
   LONG      lErrorCode;
   } WFSPINACCESS, * LPWFSPINACCESS;
```

lpsKeyName
Specifies the name of the key that caused the error.

lErrorCode
Specifies the type of illegal key access that occurred. Possible values are:

Value	Meaning
WFS_ERR_PIN_KEYNOTFOUND	The specified key was not loaded.
WFS_ERR_PIN_KEYNOVALUE	The specified key is not loaded.
WFS_ERR_PIN_USEVIOLATION	The specified use is not supported by this key.

Comments None.

6. C - Header File

```
/******************************************************************************
*                                                                             *
*xfspin.h XFS - Personal Identification Number Keypad (PIN) definitions       *
*                                                                             *
*            Version 2.00  (11/11/96)                                         *
*                                                                             *
******************************************************************************/

#ifndef __INC_XFSPIN__H
#define __INC_XFSPIN__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

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


/* values of WFSPINCAPS.wClass */

#define WFS_SERVICE_CLASS_PIN             (4)
#define WFS_SERVICE_CLASS_VERSION_PIN     (0x0002) /* Version 2.00 */
#define WFS_SERVICE_CLASS_NAME_PIN        "PIN"

#define PIN_SERVICE_OFFSET                (WFS_SERVICE_CLASS_PIN * 100)

/* PIN Info Commands */

#define WFS_INF_PIN_STATUS                (PIN_SERVICE_OFFSET + 1)
#define WFS_INF_PIN_CAPABILITIES          (PIN_SERVICE_OFFSET + 2)
#define WFS_INF_PIN_KEY_DETAIL            (PIN_SERVICE_OFFSET + 4)
#define WFS_INF_PIN_FUNCKEY_DETAIL        (PIN_SERVICE_OFFSET + 5)

/* PIN Command Verbs */

#define WFS_CMD_PIN_CRYPT                 (PIN_SERVICE_OFFSET + 1)
#define WFS_CMD_PIN_IMPORT_KEY            (PIN_SERVICE_OFFSET + 3)
#define WFS_CMD_PIN_GET_PIN               (PIN_SERVICE_OFFSET + 5)
#define WFS_CMD_PIN_GET_PINBLOCK          (PIN_SERVICE_OFFSET + 7)
#define WFS_CMD_PIN_GET_DATA              (PIN_SERVICE_OFFSET + 8)
#define WFS_CMD_PIN_INITIALIZATION        (PIN_SERVICE_OFFSET + 9)
#define WFS_CMD_PIN_LOCAL_DES             (PIN_SERVICE_OFFSET + 10)
#define WFS_CMD_PIN_LOCAL_EUROCHEQUE      (PIN_SERVICE_OFFSET + 11)
#define WFS_CMD_PIN_LOCAL_VISA            (PIN_SERVICE_OFFSET + 12)
#define WFS_CMD_PIN_CREATE_OFFSET         (PIN_SERVICE_OFFSET + 13)
#define WFS_CMD_PIN_DERIVE_KEY            (PIN_SERVICE_OFFSET + 14)
#define WFS_CMD_PIN_PRESENT_IDC           (PIN_SERVICE_OFFSET + 15)

/* PIN Messages */

#define WFS_EXEE_PIN_KEY                  (PIN_SERVICE_OFFSET + 1)
#define WFS_SRVE_PIN_INITIALIZED          (PIN_SERVICE_OFFSET + 2)
#define WFS_SRVE_PIN_ILLEGAL_KEY_ACCESS   (PIN_SERVICE_OFFSET + 3)

/* values of WFSPINSTATUS.fwDevice */

#define WFS_PIN_DEVONLINE                 WFS_STAT_DEVONLINE
#define WFS_PIN_DEVOFFLINE                WFS_STAT_DEVOFFLINE
#define WFS_PIN_DEVPOWEROFF               WFS_STAT_DEVPOWEROFF
#define WFS_PIN_DEVBUSY                   WFS_STAT_DEVBUSY
#define WFS_PIN_DEVNODEVICE               WFS_STAT_DEVNODEVICE
#define WFS_PIN_DEVHWERROR                WFS_STAT_DEVHWERROR
#define WFS_PIN_DEVUSERERROR              WFS_STAT_DEVUSERERROR
```

```
/* values of WFSPINSTATUS.fwEncStat */

#define WFS_PIN_ENCREADY                (0)
#define WFS_PIN_ENCNOTREADY             (1)
#define WFS_PIN_ENCNOTINITIALIZED       (2)
#define WFS_PIN_ENCBUSY                 (3)
#define WFS_PIN_ENCUNDEFINED            (4)
#define WFS_PIN_ENCINITIALIZED          (5)

/* values of WFSPINCAPS.wType */

#define WFS_PIN_TYPEEPP                 (0x0001)
#define WFS_PIN_TYPEEDM                 (0x0002)

/* values of WFSPINCAPS.fwAlgorithms, WFSPINCRYPT.wAlgorithm */

#define WFS_PIN_CRYPTDESECB             (0x0001)
#define WFS_PIN_CRYPTDESCBC             (0x0002)
#define WFS_PIN_CRYPTDESCFB             (0x0004)
#define WFS_PIN_CRYPTRSA                (0x0008)
#define WFS_PIN_CRYPTECMA               (0x0010)
#define WFS_PIN_CRYPTDESMAC             (0x0020)
#define WFS_PIN_CRYPTTRIDESECB          (0x0040)
#define WFS_PIN_CRYPTTRIDESCBC          (0x0080)
#define WFS_PIN_CRYPTTRIDESCFB          (0x0100)
#define WFS_PIN_CRYPTTRIDESMAC          (0x0200)

/* values of WFSPINCAPS.fwPinFormats */

#define WFS_PIN_FORM3624                (0x0001)
#define WFS_PIN_FORMANSI                (0x0002)
#define WFS_PIN_FORMISO0                (0x0004)
#define WFS_PIN_FORMISO1                (0x0008)
#define WFS_PIN_FORMECI2                (0x0010)
#define WFS_PIN_FORMECI3                (0x0020)
#define WFS_PIN_FORMVISA                WFS_PIN_FORMECI3
#define WFS_PIN_FORMDIEBOLD             (0x0080)
#define WFS_PIN_FORMDIEBOLDCO           (0x0100)

/* values of WFSPINCAPS.fwDerivationAlgorithms */

#define WFS_PIN_CHIP_ZKA                (0x0001)

/* values of WFSPINCAPS.fwPresentationAlgorithms */

#define WFS_PIN_PRESENT_CLEAR           (0x0001)

/* values of WFSPINCAPS.fwDisplay */

#define WFS_PIN_DISPNONE                (1)
#define WFS_PIN_DISPLEDTHROUGH          (2)
#define WFS_PIN_DISPDISPLAY             (3)

/* values of WFSPINCAPS.fwIDKey */

#define WFS_PIN_IDKEYINITIALIZATION     (0x0001)
#define WFS_PIN_IDKEYIMPORT             (0x0002)

/* values of WFSPINCAPS.fwValidationAlgorithms */

#define WFS_PIN_DES                     (0x0001)
#define WFS_PIN_EUROCHEQUE              (0x0002)
#define WFS_PIN_VISA                    (0x0004)
#define WFS_PIN_DES_OFFSET              (0x0008)

/* values of WFSPINKEYDETAIL.fwUse */

#define WFS_PIN_USECRYPT                (0x0001)
#define WFS_PIN_USEFUNCTION             (0x0002)
#define WFS_PIN_USEMACING               (0x0004)
#define WFS_PIN_USEKEYENCKEY            (0x0020)
#define WFS_PIN_USENODUPLICATE          (0x0040)
#define WFS_PIN_USESVENCKEY             (0x0080)
```

```
/* values of WFSPINFUNCKEYDETAIL.ulFuncMask */

#define WFS_PIN_FK_0                    (0x00000001)
#define WFS_PIN_FK_1                    (0x00000002)
#define WFS_PIN_FK_2                    (0x00000004)
#define WFS_PIN_FK_3                    (0x00000008)
#define WFS_PIN_FK_4                    (0x00000010)
#define WFS_PIN_FK_5                    (0x00000020)
#define WFS_PIN_FK_6                    (0x00000040)
#define WFS_PIN_FK_7                    (0x00000080)
#define WFS_PIN_FK_8                    (0x00000100)
#define WFS_PIN_FK_9                    (0x00000200)
#define WFS_PIN_FK_ENTER                (0x00000400)
#define WFS_PIN_FK_CANCEL               (0x00000800)
#define WFS_PIN_FK_CLEAR                (0x00001000)
#define WFS_PIN_FK_BACKSPACE            (0x00002000)
#define WFS_PIN_FK_HELP                 (0x00004000)
#define WFS_PIN_FK_DECPOINT             (0x00008000)
#define WFS_PIN_FK_00                   (0x00010000)
#define WFS_PIN_FK_000                  (0x00020000)
#define WFS_PIN_FK_RES1                 (0x00040000)
#define WFS_PIN_FK_RES2                 (0x00080000)
#define WFS_PIN_FK_RES3                 (0x00100000)
#define WFS_PIN_FK_RES4                 (0x00200000)
#define WFS_PIN_FK_RES5                 (0x00400000)
#define WFS_PIN_FK_RES6                 (0x00800000)
#define WFS_PIN_FK_RES7                 (0x01000000)
#define WFS_PIN_FK_RES8                 (0x02000000)
#define WFS_PIN_FK_OEM1                 (0x04000000)
#define WFS_PIN_FK_OEM2                 (0x08000000)
#define WFS_PIN_FK_OEM3                 (0x10000000)
#define WFS_PIN_FK_OEM4                 (0x20000000)
#define WFS_PIN_FK_OEM5                 (0x40000000)
#define WFS_PIN_FK_OEM6                 (0x80000000)

/* values of WFSPINFUNCKEY.ulFDK */

#define WFS_PIN_FK_FDK01                (0x00000001)
#define WFS_PIN_FK_FDK02                (0x00000002)
#define WFS_PIN_FK_FDK03                (0x00000004)
#define WFS_PIN_FK_FDK04                (0x00000008)
#define WFS_PIN_FK_FDK05                (0x00000010)
#define WFS_PIN_FK_FDK06                (0x00000020)
#define WFS_PIN_FK_FDK07                (0x00000040)
#define WFS_PIN_FK_FDK08                (0x00000080)
#define WFS_PIN_FK_FDK09                (0x00000100)
#define WFS_PIN_FK_FDK10                (0x00000200)
#define WFS_PIN_FK_FDK11                (0x00000400)
#define WFS_PIN_FK_FDK12                (0x00000800)
#define WFS_PIN_FK_FDK13                (0x00001000)
#define WFS_PIN_FK_FDK14                (0x00002000)
#define WFS_PIN_FK_FDK15                (0x00004000)
#define WFS_PIN_FK_FDK16                (0x00008000)
#define WFS_PIN_FK_FDK17                (0x00010000)
#define WFS_PIN_FK_FDK18                (0x00020000)
#define WFS_PIN_FK_FDK19                (0x00040000)
#define WFS_PIN_FK_FDK20                (0x00080000)
#define WFS_PIN_FK_FDK21                (0x00100000)
#define WFS_PIN_FK_FDK22                (0x00200000)
#define WFS_PIN_FK_FDK23                (0x00400000)
#define WFS_PIN_FK_FDK24                (0x00800000)
#define WFS_PIN_FK_FDK25                (0x01000000)
#define WFS_PIN_FK_FDK26                (0x02000000)
#define WFS_PIN_FK_FDK27                (0x04000000)
#define WFS_PIN_FK_FDK28                (0x08000000)
#define WFS_PIN_FK_FDK29                (0x10000000)
#define WFS_PIN_FK_FDK30                (0x20000000)
#define WFS_PIN_FK_FDK31                (0x40000000)
#define WFS_PIN_FK_FDK32                (0x80000000)

/* values of WFSPINCRYPT.wMode */

#define WFS_PIN_MODEENCRYPT             (1)
#define WFS_PIN_MODEDECRYPT             (2)
```

```
/* values of WFSPINENTRY.wCompletion */

#define WFS_PIN_COMPAUTO                (0)
#define WFS_PIN_COMPENTER               (1)
#define WFS_PIN_COMPCANCEL              (2)
#define WFS_PIN_COMPCONTINUE            (6)
#define WFS_PIN_COMPCLEAR               (7)
#define WFS_PIN_COMPBACKSPACE           (8)
#define WFS_PIN_COMPFDK                 (9)
#define WFS_PIN_COMPHELP                (10)
#define WFS_PIN_COMPFK                  (11)


/* XFS PIN Errors */

#define WFS_ERR_PIN_KEYNOTFOUND         (-(PIN_SERVICE_OFFSET + 0))
#define WFS_ERR_PIN_MODENOTSUPPORTED    (-(PIN_SERVICE_OFFSET + 1))
#define WFS_ERR_PIN_ACCESSDENIED        (-(PIN_SERVICE_OFFSET + 2))
#define WFS_ERR_PIN_INVALIDID           (-(PIN_SERVICE_OFFSET + 3))
#define WFS_ERR_PIN_DUPLICATEKEY        (-(PIN_SERVICE_OFFSET + 4))
#define WFS_ERR_PIN_KEYNOVALUE          (-(PIN_SERVICE_OFFSET + 6))
#define WFS_ERR_PIN_USEVIOLATION        (-(PIN_SERVICE_OFFSET + 7))
#define WFS_ERR_PIN_NOPIN               (-(PIN_SERVICE_OFFSET + 8))
#define WFS_ERR_PIN_INVALIDKEYLENGTH    (-(PIN_SERVICE_OFFSET + 9))
#define WFS_ERR_PIN_KEYINVALID          (-(PIN_SERVICE_OFFSET + 10))
#define WFS_ERR_PIN_KEYNOTSUPPORTED     (-(PIN_SERVICE_OFFSET + 11))
#define WFS_ERR_PIN_NOACTIVEKEYS        (-(PIN_SERVICE_OFFSET + 12))
#define WFS_ERR_PIN_INVALIDKEY          (-(PIN_SERVICE_OFFSET + 13))
#define WFS_ERR_PIN_NOTERMINATEKEYS     (-(PIN_SERVICE_OFFSET + 14))
#define WFS_ERR_PIN_MINIMUMLENGTH       (-(PIN_SERVICE_OFFSET + 15))
#define WFS_ERR_PIN_PROTOCOLNOTSUPP     (-(PIN_SERVICE_OFFSET + 16))
#define WFS_ERR_PIN_INVALIDDATA         (-(PIN_SERVICE_OFFSET + 17))
#define WFS_ERR_PIN_NOTALLOWED          (-(PIN_SERVICE_OFFSET + 18))


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

typedef struct _wfs_pin_status
{
    WORD                fwDevice;
    WORD                fwEncStat;
    LPSTR               lpszExtra;
} WFSPINSTATUS, * LPWFSPINSTATUS;

typedef struct _wfs_pin_caps
{
    WORD                wClass;
    WORD                fwType;
    BOOL                bCompound;
    USHORT              usKeyNum;
    WORD                fwAlgorithms;
    WORD                fwPinFormats;
    WORD                fwDerivationAlgorithms;
    WORD                fwPresentationAlgorithms;
    WORD                fwDisplay;
    BOOL                bIDConnect;
    WORD                fwIDKey;
    WORD                fwValidationAlgorithms;
    LPSTR               lpszExtra;
} WFSPINCAPS, * LPWFSPINCAPS;

typedef struct _wfs_pin_key_detail
{
    LPSTR               lpsKeyName;
    WORD                fwUse;
    BOOL                bLoaded;
} WFSPINKEYDETAIL, * LPWFSPINKEYDETAIL;

typedef struct _wfs_pin_fdk
{
    ULONG               ulFDK;
```

```
    USHORT              usXPosition;
    USHORT              usYPosition;
} WFSPINFDK, * LPWFSPINFDK;

typedef struct _wfs_pin_func_key_detail
{
    ULONG               ulFuncMask;
    USHORT              usNumberFDKs;
    LPWFSPINFDK       * lppFDKs;
} WFSPINFUNCKEYDETAIL, * LPWFSPINFUNCKEYDETAIL;

/*=================================================================*/
/* PIN Execute Command Structures */
/*=================================================================*/

typedef struct _wfs_hex_data
{
    USHORT              usLength;
    LPBYTE              lpbData;
} WFSXDATA, * LPWFSXDATA;

typedef struct _wfs_pin_crypt
{
    WORD                wMode;
    LPSTR               lpsKey;
    LPWFSXDATA          lpxKeyEncKey;
    WORD                wAlgorithm;
    LPSTR               lpsStartValueKey;
    LPWFSXDATA          lpxStartValue;
    BYTE                bPadding;
    BYTE                bCompression;
    LPWFSXDATA          lpxCryptData;
} WFSPINCRYPT, * LPWFSPINCRYPT;

typedef struct _wfs_pin_import
{
    LPSTR               lpsKey;
    LPSTR               lpsEncKey;
    LPWFSXDATA          lpxIdent;
    LPWFSXDATA          lpxValue;
    WORD                fwUse;
} WFSPINIMPORT, * LPWFSPINIMPORT;

typedef struct _wfs_pin_derive
{
    WORD                wDerivationAlgorithm;
    LPSTR               lpsKey;
    LPSTR               lpsKeyGenKey;
    LPSTR               lpsStartValueKey;
    LPWFSXDATA          lpxStartValue;
    BYTE                bPadding;
    LPWFSXDATA          lpxInputData;
    LPWFSXDATA          lpxIdent;
 } WFSPINDERIVE, * LPWFSPINDERIVE;

typedef struct _wfs_pin_getpin
{
    USHORT              usMinLen;
    USHORT              usMaxLen;
    BOOL                bAutoEnd;
    CHAR                cEcho;
    ULONG               ulActiveFDKs;
    ULONG               ulActiveKeys;
    ULONG               ulTerminateFDKs;
    ULONG               ulTerminateKeys;
} WFSPINGETPIN, * LPWFSPINGETPIN;

typedef struct _wfs_pin_entry
{
    USHORT              usDigits;
    WORD                wCompletion;
} WFSPINENTRY, * LPWFSPINENTRY;

typedef struct _wfs_pin_local_des
```

```
{
    LPSTR               lpsValidationData;
    LPSTR               lpsOffset;
    BYTE                bPadding;
    USHORT              usMaxPIN;
    USHORT              usValDigits;
    BOOL                bNoLeadingZero;
    LPSTR               lpsKey;
    LPWFSXDATA          lpxKeyEncKey;
    LPSTR               lpsDecTable;
} WFSPINLOCALDES, * LPWFSPINLOCALDES;

typedef struct _wfs_pin_create_offset
{
    LPSTR               lpsValidationData;
    BYTE                bPadding;
    USHORT              usMaxPIN;
    USHORT              usValDigits;
    LPSTR               lpsKey;
    LPWFSXDATA          lpxKeyEncKey;
    LPSTR               lpsDecTable;
} WFSPINCREATEOFFSET, * LPWFSPINCREATEOFFSET;

typedef struct _wfs_pin_local_eurocheque
{
    LPSTR               lpsEurochequeData;
    LPSTR               lpsPVV;
    WORD                wFirstEncDigits;
    WORD                wFirstEncOffset;
    WORD                wPVVDigits;
    WORD                wPVVOffset;
    LPSTR               lpsKey;
    LPWFSXDATA          lpxKeyEncKey;
    LPSTR               lpsDecTable;
} WFSPINLOCALEUROCHEQUE, * LPWFSPINLOCALEUROCHEQUE;

typedef struct _wfs_pin_local_visa
{
    LPSTR               lpsPAN;
    LPSTR               lpsPVV;
    WORD                wPVVDigits;
    LPSTR               lpsKey;
    LPWFSXDATA          lpxKeyEncKey;
} WFSPINLOCALVISA, * LPWFSPINLOCALVISA;

typedef struct _wfs_pin_presentidc
{
    WORD                wPresentAlgorithm;
    WORD                wChipProtocol;
    ULONG               ulChipDataLength;
    LPBYTE              lpbChipData;
    LPVOID              lpAlgorithmData;
} WFSPINPRESENTIDC, * LPWFSPINPRESENTIDC;

typedef struct _wfs_pin_present_result
{
    WORD                wChipProtocol;
    ULONG               ulChipDataLength;
    LPBYTE              lpbChipData;
} WFSPINPRESENTRESULT, * LPWFSPINPRESENTRESULT;


typedef struct _wfs_pin_presentclear
{
    ULONG               ulPINPointer;
    USHORT              usPINOffset;
} WFSPINPRESENTCLEAR, * LPWFSPINPRESENTCLEAR;

typedef struct _wfs_pin_block
{
    LPSTR               lpsCustomerData;
    LPSTR               lpsXORData;
    BYTE                bPadding;
    WORD                wFormat;
    LPSTR               lpsKey;
```

```
    LPSTR               lpsKeyEncKey;
} WFSPINBLOCK, * LPWFSPINBLOCK;

typedef struct _wfs_pin_getdata
{
    USHORT              usMaxLen;
    BOOL                bAutoEnd;
    ULONG               ulActiveFDKs;
    ULONG               ulActiveKeys;
    ULONG               ulTerminateFDKs;
    ULONG               ulTerminateKeys;
} WFSPINGETDATA, * LPWFSPINGETDATA;

typedef struct _wfs_pin_data
{
    LPSTR               lpsData;
    WORD                wCompletion;
} WFSPINDATA, * LPWFSPINDATA;

typedef struct _wfs_pin_init
{
    LPWFSXDATA          lpxIdent;
    LPWFSXDATA          lpxKey;
} WFSPININIT, * LPWFSPININIT;


/*=====================================================================*/
/* PIN Message Structures */
/*=====================================================================*/

typedef struct _wfs_pin_key
{
    WORD                wCompletion;
    ULONG               ulDigit;
} WFSPINKEY, * LPWFSPINKEY;

typedef struct _wfs_pin_access
    {
    LPSTR               lpsKeyName;
    LONG                lErrorCode;
    } WFSPINACCESS, * LPWFSPINACCESS;


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

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

#endif    /* __INC_XFSPIN__H */
```