CEN

WORKSHOP

AGREEMENT

CWA 13449-9

December 1998

ICS 35.200;35.240.40

English version

Extensions for Financial Services (XFS) interface specification - Part 9: Text Terminal Unit Device Class Interface - Programmer's Interface

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

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

Ref. No. CWA 13449-9:1998 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 9 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 specifiction is to standardize function codes and structures for the broadest variety of services. For example, using the **WFSExecute** function, the commands to read data from various services are as similar as possible to each other in their syntax and data structures.

In general, the specific command set for a service class is defined as a superset of the specific capabilities likely to be provided by the developers of the services of that class; thus any particular device will normally support only a subset of the defined command set.
There are three cases in which a service provider may receive a service-specific command that it does not support:

- The requested capability is defined for the class of service providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability is ***not*** considered to be fundamental to the service. In this case, the service provider returns a successful completion, but does no operation. An example would be a request from an application to turn on a control indicator on a passbook printer; the service provider recognizes the command, but since the passbook printer it is managing does not include that indicator, the service provider does no operation and returns a successful completion to the application.
- The requested capability is defined for the class of service providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability ***is*** considered to be fundamental to the service. In this case, a WFS_ERR_UNSUPP_COMMAND error is returned to the calling application. An example would be a request from an application to a cash dispenser to dispense coins; the service provider recognizes the command but, since the cash dispenser it is managing dispenses only notes, returns this error.
- The requested capability is ***not*** defined for the class of service providers by the XFS specification. In this case, a WFS_ERR_INVALID_COMMAND error is returned to the calling application.

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

2. Text Terminal Unit

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

This section describes the functions provided by a generic Text Terminal Unit (TTU) service. A Text Terminal Unit is a text i/o device, which applies both to ATM operator panels and to displays incorporated in devices such as PIN pads and printers. This service allows for the following categories of functions:

- Forms oriented input and output
- Direct display output
- Keyboard input
- LED settings and control

3. Info Commands

3.1 WFS_INF_TTU_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

```
LPWFSTTUSTATUS      lpStatus;

typedef struct _wfs_ttu_status
   {
   WORD          fwDevice;
   WORD          wKeyboard;
   WORD          wKeyLock;
   WORD          wLEDs [WFS_TTU_LEDS_MAX];
   WORD          wDisplaySizeX;
   WORD          wDisplaySizeY;
   LPSTR         lpszExtra;
   } WFSTTUSTATUS, * LPWFSTTUSTATUS;
```

fwDevice
Specifies the state of the text terminal unit as one of the following flags:

Value	Meaning
WFS_TTU_DEVONLINE	The device is on-line. The device is present and operational (i.e. not busy processing a request and not having a hardware error).
WFS_TTU_DEVOFFLINE	The device is off-line. The device is present and powered on but it is not operational (e.g. a switch may have been used to change it to an off-line state).
WFS_TTU_DEVPOWEROFF	The device is powered off. The device is present, but is currently powered off.
WFS_TTU_DEVBUSY	The device is busy processing a request. The device is present and an EXECUTE request is currently being processed.
WFS_TTU_DEVNODEVICE	There is no device connected.
WFS_TTU_DEVHWERROR	The device is inoperable due to a hardware error. The device is present but a hardware fault prevents it from being used.
WFS_TTU_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 (WFS_TTU_DEVONLINE) or a permanent error condition has occurred (WFS_TTU_DEVHWERROR).

wKeyboard
Specifies the state of the keyboard in the text terminal unit as one of the following flags:

Value	Meaning
WFS_TTU_KBDON	The keyboard is activated.
WFS_TTU_KBDOFF	The keyboard is not activated.
WFS_TTU_KBDNA	The keyboard is not available.

wKeyLock
Specifies the state of the keyboard lock of the text terminal unit as one of the following flags:

Value	Meaning
WFS_TTU_KBDLOCKON	The keyboard lock switch is activated.
WFS_TTU_KBDLOCKOFF	The keyboard lock switch is not activated.

WFS_TTU_KBDLOCKNA	The keyboard lock switch is not available.

wLEDs [WFS_TTU_LEDS_MAX]
Specifies the state of the LEDs. The maximum guidance light index is WFS_TTU_LEDS_MAX. The number of available LEDs can be retrieved with the WFS_INF_TTU_CAPABILITIES info command. All member elements in this array are specified as one of the following flags:

Value	Meaning
WFS_TTU_LEDNA	The status is not available.
WFS_TTU_LEDOFF	The LED is turned **off**.
WFS_TTU_LEDSLOWFLASH	The LED is **blinking slowly**.
WFS_TTU_LEDMEDIUMFLASH	The LED is **blinking medium frequency**.
WFS_TTU_LEDQUICKFLASH	The LED is **blinking quickly**.
WFS_TTU_LEDCONTINUOUS	The light is turned on **continuous** (steady).

wDisplaySizeX
Specifies the horizontal size of the display of the text terminal unit (the number of columns that can be displayed).

wDisplaySizeY
Specifies the vertical size of the display of the text terminal unit (the number of rows that can be displayed).

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 extensible 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_TTU_CAPABILITIES

Description This command is used to retrieve the capabilities of the text terminal unit.

Input Param None.

Output Param `LPWFSTTUCAPS lpCaps;`

```
typedef struct _wfs_ttu_caps
   {
   WORD                    wClass;
   WORD                    fwType;
   LPWFSTTURESOLUTION *    lppResolutions;
   WORD                    wNumOfLEDs;
   WORD                    fwKeys;
   BOOL                    bKeyLock;
   BOOL                    bDisplayLight;
   BOOL                    bCursor;
   BOOL                    bForms;
   LPSTR                   lpszExtra;
   } WFSTTUCAPS, * LPWFSTTUCAPS;
```

wClass
Specifies the logical service class, value is:
WFS_SERVICE_CLASS_TTU

fwType
Specifies the type of the text terminal unit as one of the following flags:

Value	Meaning
WFS_TTU_FIXED	The text terminal unit is a fixed device.
WFS_TTU_REMOVABLE	The text terminal unit is a removable device.

lppResolutions
Pointer to a NULL terminated array of pointers WFSTTURESOLUTION structures. Specifies the resolutions supported by the physical display device. (For a definition of WFSTTURESOLUTION see command WFS_CMD_TTU_SET_RESOLUTION).

wNumOfLEDs
Specifies the number of LEDs available in this text terminal unit.

fwKeys
Specifies which types of keys the key pad of the text terminal unit supports as a combination of the following flags:

Value	Meaning
WFS_TTU_KEYNUMERIC	The text terminal unit has keys for numeric values.
WFS_TTU_KEYHEXADECIMAL	The text terminal unit has keys for hexadecimal values.
WFS_TTU_KEYALPHANUMERIC	The text terminal unit has keys for alphanumeric values.

bKeyLock
Specifies whether the text terminal unit has a key lock switch. The value can be either FALSE (not available) or TRUE (available).

bDisplayLight
Specifies whether the text terminal unit has a display light. The value can be either FALSE (not available) or TRUE (available).

bCursor
Specifies whether the text terminal unit display supports a cursor. The value can be either FALSE (not available) or TRUE (available).

bForms
Specifies whether the text terminal unit service supports forms oriented input and output. The value can be either FALSE (not available) or TRUE (available).

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 extensible 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.

3.3 WFS_INF_TTU_FORM_LIST

Description This command is used to retrieve the list of forms available on the device.

Input Param None.

Output Param
```
LPSTR         lpszFormList;
```

lpszFormList
Pointer to a list of null-terminated form names, with the final name terminating with two null characters.

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

Comments None.

3.4 WFS_INF_TTU_QUERY_FORM

Description This command is used to retrieve details of the definition of a specified form.

Input Param

```
LPSTR        lpszFormName;
```

lpszFormName
Points to the null-terminated form name on which to retrieve details.

Output Param

```
LPWFSTTUFRMHEADER lpFrmHeader;

typedef struct _wfs_ttu_frm_header
   {
   LPSTR     lpszFormName;
   WORD      wWidth;
   WORD      wHeight;
   WORD      wVersionMajor;
   WORD      wVersionMinor;
   LPSTR     lpszFields;
   } WFSTTUFRMHEADER, * LPWFSTTUFRMHEADER;
```

lpszFormName
Specifies the null-terminated name of the form.

wWidth
Specifies the width of the form in columns.

wHeight
Specifies the height of the form in rows.

wVersionMajor
Specifies the major version of the form.

wVersionMinor
Specifies the minor version of the form.

lpszFields
Pointer to a list of null-terminated field names, with the final name terminating with two null characters.

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

Value	Meaning
WFS_ERR_TTU_FORMNOTFOUND	The specified form cannot be found.
WFS_ERR_TTU_FORMINVALID	The specified form is invalid.

Comments None.

3.5 WFS_INF_TTU_QUERY_FIELD

Description This command is used to retrieve details of the definition of a single or all fields on a specified form.

Input Param

```
LPWFSTTUQUERYFIELD  lpQueryField;
```

```
typedef struct _wfs_ttu_query_field
   {
   LPSTR            lpszFormName;
   LPSTR            lpszFieldName;
   } WFSTTUQUERYFIELD, * LPWFSTTUQUERYFIELD;
```

lpszFormName
Pointer to the null-terminated form name.

lpszFieldName
Pointer to the null-terminated name of the field about which to retrieve details. If this value is NULL, then retrieve details for all fields on the form.

Output Param `LPWFSTTUFRMFIELD *  lppFields;`

lppFields
Pointer to a null-terminated array of pointers to field definition structures:

```
typedef struct _wfs_ttu_frm_field
   {
   LPSTR      lpszFieldName;
   WORD       fwType;
   WORD       fwClass;
   WORD       fwAccess;
   WORD       fwOverflow;
   LPSTR      lpszFormat;
   } WFSTTUFRMFIELD, * LPWFSTTUFRMFIELD;
```

lpszFieldName
Pointer to the null-terminated field name.

fwType
Specifies the type of field and can be one of the following:

Value	Meaning
WFS_TTU_FIELDTEXT	A text field.
WFS_TTU_FIELDINVISIBLE	An invisible text field.
WFS_TTU_FIELDPASSWORD	A password field, input is echoed as ‘*’.

fwClass
Specifies the class of the field and can be one of the following:

Value	Meaning
WFS_TTU_CLASSSTATIC	The field data cannot be set by the application.
WFS_TTU_CLASSOPTIONAL	The field data can be set by the application.
WFS_TTU_CLASSREQUIRED	The field data must be set by the application.

fwAccess
Specifies whether the field is to be used for input, output, or both and can be a combination of the following bit-flags:

Value	Meaning
WFS_TTU_ACCESSREAD	The field is used for input from the physical device.
WFS_TTU_ACCESSWRITE	The field is used for output to the physical device.

fwOverflow
Specifies how an overflow of field data should be handled and can be one of the following:

Value	Meaning
WFS_TTU_OVFTERMINATE	Return an error and terminate display of the form.
WFS_TTU_OVFTRUNCATE	Truncate the field data to fit in the field.
WFS_TTU_OVFOVERWRITE	Print the field data beyond the extents of the field boundary.

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

Value	Meaning
WFS_ERR_TTU_FORMNOTFOUND	The specified form cannot be found.
WFS_ERR_TTU_FORMINVALID	The specified form is invalid.
WFS_ERR_TTU_FIELDNOTFOUND	The specified field cannot be found.
WFS_ERR_TTU_FIELDINVALID	The specified field is invalid.

Comments None.

4. Execute Commands

4.1 WFS_CMD_TTU_BEEP

Description This command is used to beep at the text terminal unit.

Input Param

```
LPWORD       lpwBeep;
```

lpwBeep
Specifies whether the beeper should be turned on or off. Specified as one of the following flags of type A and B, or as WFS_TTU_BEEPCONTINUOUS in combination with one of the flags of type B:

Value	Meaning	Type
WFS_TTU_BEEPOFF	The beeper is turned off.	A
WFS_TTU_BEEPKEYPRESS	The beeper sounds a key click signal.	B
WFS_TTU_BEEPEXCLAMATION	The beeper sounds a exclamation signal.	B
WFS_TTU_BEEPWARNING	The beeper sounds a warning signal.	B
WFS_TTU_BEEPERROR	The beeper sounds a error signal.	B
WFS_TTU_BEEPCRITICAL	The beeper sounds a critical error signal.	B
WFS_TTU_BEEPCONTINUOUS	The beeper sound is turned on continuously.	C

Output Param None.

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

Events There are no additional events generated by this command.

Comments None.

4.2 WFS_CMD_TTU_CLEARSCREEN

Description This command clears the specified area of the text terminal unit screen. The cursor is positioned to the upper left corner of the cleared area.

Input Param

```
LPWFSTTUCLEARSCREEN lpClearScreen;

struct _wfs_ttu_clear_screen
   {
   WORD          wPositionX;
   WORD          wPositionY;
   WORD          wWidth;
   WORD          wHeight;
   } WFSTTUCLEARSCREEN, * LPWFSTTUCLEARSCREEN;
```

wPositionX
Specifies the horizontal position of the area to be cleared.

wPositionY
Specifies the vertical position of the area to be cleared.

wWidth
Specifies the width of the area to be cleared.

wHeight
Specifies the height of the area to be cleared.

Output Param None.

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

Events There are no additional events generated by this command.

Comments If the input parameter is NULL, the whole screen will be cleared.

4.3 WFS_CMD_TTU_DISPLIGHT

Description This command is used to switch the lighting of the text terminal unit on or off.

Input Param `LPWFSTTUDISPLIGHT   lpDispLight;`

```
typedef struct _wfs_ttu_disp_light
    {
    BOOL      bMode;
    } WFSTTUDISPLIGHT, * LPWFSTTUDISPLIGHT;
```

bMode
Specifies whether the lighting of the text terminal unit is switched on (TRUE) or off (FALSE).

Output Param None.

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

Events There are no additional events generated by this command.

Comments None.

4.4 WFS_CMD_TTU_SET_LED

Description This command is used to set the status of the LEDs.

Input Param `LPWFSTTUSETLEDS     lpSetLEDs;`

```
typedef struct _wfs_ttu_set_leds
    {
    WORD         wLED;
    WORD         fwCommand;
    } WFSTTUSETLEDS, * LPWFSTTUSETLEDS;
```

wLED
Specifies the index of the LED to set.

fwCommand
Specifies the state of the LED, as one of the following flags:

Value	Meaning
WFS_TTU_LEDOFF	The LED is turned off.
WFS_TTU_LEDSLOWFLASH	The LED is set to flash slowly.
WFS_TTU_LEDMEDIUMFLASH	The LED is blinking medium frequency.
WFS_TTU_LEDQUICKFLASH	The LED is set to flash quickly.
WFS_TTU_LEDCONTINUOUS	The LED is turned on continuously (steady).

Output Param None.

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

Value	Meaning
WFS_ERR_TTU_INVALIDLED	An attempt to set a LED to a new value was invalid because the LED does not exist.

Events There are no additional events generated by this command.

Comments None.

4.5 WFS_CMD_TTU_SET_RESOLUTION

Description This command is used to set the resolution of the display.

Input Param `LPWFSTTURESOLUTION  lpResolution;`

```
typedef struct _wfs_ttu_resolution
   {
   WORD            wSizeX;
   WORD            wSizeY;
   } WFSTTURESOLUTION, * LPWFSTTURESOLUTION;
```

wSizeX
Specifies the horizontal size of the display of the text terminal unit (the number of columns that can be displayed)

wSizeY
Specifies the vertical size of the display of the text terminal unit (the number of rows that can be displayed)

Output Param None.

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

Value	Meaning
WFS_ERR_TTU_RESNOTSUPP	The specified resolution is not supported by the display.

Events There are no additional events generated by this command.

Comments None.

4.6 WFS_CMD_TTU_DISPLAY_FORM

Description This command is used to display a form by merging the supplied variable field data with the defined form and field data specified in the form.

Input Param `LPWFSTTUDISPLAYFORM lpDisplayform;`

```
typedef struct _wfs_ttu_display_form
   {
   LPSTR     lpszFormName;
   BOOL      bClearScreen;
   LPSTR     lpszFields;
   } WFSTTUDISPLAYFORM, * LPWFSTTUDISPLAYFORM;
```

lpszFormName
Pointer to the null-terminated form name.

bClearScreen
Specifies whether the screen is cleared before displaying the form (TRUE) or not (FALSE).

lpszFields
Pointer to a series of "<FieldName>=<FieldValue>" strings, where each string is null-terminated with the final string terminating with two null characters.

Output Param None.

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

Value	Meaning
WFS_ERR_TTU_FORMNOTFOUND	The specified form definition cannot be found.
WFS_ERR_TTU_FORMINVALID	The specified form definition is invalid.
WFS_ERR_TTU_MEDIAOVERFLOW	The form overflowed the media.
WFS_ERR_TTU_FIELDSPECFAILURE	The syntax of the *lpszFields* member is invalid.
WFS_ERR_TTU_FIELDERROR	An error occurred while processing a field, causing termination of the display request

Events There are no additional events generated by this command.

Comments None.

4.7 WFS_CMD_TTU_READ_FORM

Description This command is used to read data from input fields on the specified form.

Input Param `LPWFSTTUREADFORM lpReadForm;`

```
typedef struct _wfs_ttu_read_form
   {
   LPSTR     lpszFormName;
   LPSTR     lpszFieldNames;
   } WFSTTUREADFORM, * LPWFSTTUREADFORM;
```

lpszFormName
Pointer to the null-terminated name of the form.

lpszFieldNames
Pointer to a list of null-terminated field names from which to read input data, with the final name terminating with two null characters. If this value is NULL, then read data from all input fields on the form.

Output Param `LPSTR  lpszFields;`

lpszFields
Pointer to a series of "<FieldName>=<FieldValue>" strings, where each string is null-terminated with the final string terminating with two null characters.

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

Value	Meaning
WFS_ERR_TTU_FORMNOTFOUND	The specified form cannot be found.
WFS_ERR_TTU_FORMINVALID	The specified form definition is invalid.
WFS_ERR_TTU_FIELDSPECFAILURE	The syntax of the *lpszFieldNames* member is invalid.
WFS_ERR_TTU_KEYCANCELED	The read operation was terminated by pressing the <CANCEL>-key.

Events There are no additional events generated by this command.

Comments None.

4.8 WFS_CMD_TTU_WRITE

Description This command displays the specified text on the display of the text terminal unit .

Input Param `LPWFSTTUWRITE lpWrite;`

```
typedef struct _wfs_ttu_write
   {
   WORD         fwMode;
   WORD         wPosX;
   WORD         wPosY;
   WORD         fwTextAttr;
   LPSTR        lpsText;
   } WFSTTUWRITE, * LPWFSTTUWRITE;
```

fwMode
Specifies whether the position of the output is absolute or relative to the current cursor position. Possible values are:

Value	Meaning
WFS_TTU_POSRELATIVE	The output is positioned relative to the current cursor position.
WFS_TTU_POSABSOLUTE	The output is positioned absolute at the position specified in *wPosX* and *wPosY*.

wPosX
Specifies the horizontal position, if *fwMode* is set to WFS_TTU_POSABSOLUTE. Or an offset relative to the current cursor position, if *fwMode* is set to WFS_TTU_POSRELATIVE.

wPosY
Specifies the vertical position, if *fwMode* is set to WFS_TTU_POSABSOLUTE. Or an offset relative to the current cursor position, if *fwMode* is set to WFS_TTU_POSRELATIVE.

fwTextAttr
Specifies the text attributes used for displaying the text as a combination of the following flags:

Value	Meaning
WFS_TTU_TEXTUNDERLINED	The displayed text will be underlined.
WFS_TTU_TEXTINVERTED	The displayed text will be inverted.
WFS_TTU_TEXTFLASH	The displayed text will be flashing.

lpsText
Specifies the text that will be displayed.

Output Param None.

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

Events There are no additional events generated by this command.

Comments None.

4.9 WFS_CMD_TTU_READ

Description This command activates the keyboard of the text terminal unit for input of the specified number of characters. Depending on the specified flush mode the input buffer is cleared.

Input Param `LPWFSTTUREAD lpRead;`

```
typedef struct _wfs_ttu_read
   {
   WORD         wNumOfChars;
   WORD         fwMode;
   WORD         wPosX;
   WORD         wPosY;
   WORD         fwEchoMode;
   WORD         fwEchoAttr;
   WORD         wKeys;
   BOOL         bCursor;
   BOOL         bFlush;
   BOOL         bAutoEnd;
   } WFSTTUREAD, * LPWFSTTUREAD;
```

wNumOfChars
Specifies the number of characters that will be read from the text terminal unit key pad.

fwMode
Specifies where the cursor is positioned for the read operation. Possible values are:

Value	Meaning
WFS_TTU_POSRELATIVE	The cursor is positioned relative to the current cursor position.
WFS_TTU_POSABSOLUTE	The cursor is positioned absolute at the position specified in *wPosX* and *wPosY*.

wPosX
Specifies the horizontal position, if *fwMode* is set to WFS_TTU_POSABSOLUTE. Or an offset relative to the current cursor position, if *fwMode* is set to WFS_TTU_POSRELATIVE.

wPosY
Specifies the vertical position, if *fwMode* is set to WFS_TTU_POSABSOLUTE. Or an offset relative to the current cursor position, if *fwMode* is set to WFS_TTU_POSRELATIVE.

fwEchoMode
Specifies how the user input is echoed to the screen as one of the following flags:

Value	Meaning
WFS_TTU_ECHOTEXT	The user input is echoed to the screen.
WFS_TTU_ECHOINVISIBLE	The user input is not echoed to the screen.
WFS_TTU_ECHOPASSWORD	The keys entered by the user are echoed as the replace character on the screen.

fwEchoAttr
Specifies the text attributes with which the user input is echoed to the screen as a combination of the following flags:

Value	Meaning
WFS_TTU_TEXTUNDERLINED	The displayed text will be underlined.
WFS_TTU_TEXTINVERTED	The displayed text will be inverted.
WFS_TTU_TEXTFLASH	The displayed text will be flashing.

wKeys
Specifies the keys which will be accepted as input to this command as on of the following flags:

Value	Meaning
WFS_TTU_KEYNUMERIC	Accept numeric values.
WFS_TTU_KEYHEXADECIMAL	Accept hexadecimal values.
WFS_TTU_KEYALPHANUMERIC	Accept alphanumeric values.

bCursor
Specifies whether the cursor is visible (TRUE) or invisible (FALSE).

bFlush
Specifies whether the keyboard input buffer is cleared before allowing for user input (TRUE) or not (FALSE).

bAutoEnd
Specifies whether the command input is automatically ended by the Service Provider if the maximum number of digits is entered.

Output Param `LPSTR  lpszInput;`

lpszInput
Specifies a zero terminated string containing all the characters read from the text terminal unit key pad.

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

Value	Meaning
WFS_ERR_TTU_KEYCANCELED	The read operation was terminated by pressing the <CANCEL>-key.

Events There are no additional events generated by this command.

Comments None.

5. Events

There are no additional events generated by this device class.

6. Form and Field Definitions

This section outlines the format of the definitions of forms, the fields within them, and the media on which they are printed.

6.1 Definition Syntax

The syntactic rules for form, field and media definitions are as follows:

• White space	space, tab
• Line continuation	backslash (\)
• Line termination	CR, LF, CR/LF; line termination ends a "keyword section" (a keyword and its value[s])
• Keywords	must be all upper case
• Names	(field/media/font names) any case; case is preserved; service providers are case sensitive
• Strings	all strings must be enclosed in double quote characters ("); standard C escape sequences are allowed.
• Comments	start with two forward slashes (//), end at line termination

Other notes:

- If a keyword is present, all its values must be specified; default values are used only if the keyword is absent.
- Values that are character strings are marked with asterisks in the definitions below, and must be quoted as specified above.
- Fields are processed in the sequence they are defined in the form.

6.2 Form Definition

XFSFORM		*formname**	
BEGIN			
(required)	**SIZE**	*width,* *height*	Width of form Height of form
	VERSION	*major,* *minor,* *date*,* *author**	Major version number Minor version number Creation/modification date Author of form
(required)	**LANGUAGE**	*languageID*	Language used in this form – a 16 bit value (LANGID) which is a combination of a primary (10 bits) and a secondary (6 bits) language ID (This is the standard language ID in the Win32 API; standard macros support construction and decomposition of this composite ID)
	COPYRIGHT	*copyright**	Copyright entry
	TITLE	*title**	Title of form
	COMMENT	*comment**	Comment section
	[XFSFIELD	*fieldname**	One field definition (as defined in the next section) for each field in the form

	BEGIN
	. . .
	END]		

END			

6.3 Field Definition

XFSFIELD		*fieldname**	
BEGIN			
(required)	**POSITION**	*x,* *y*	Horizontal position (relative to left side of form) Vertical position (relative to top of form)
(required)	**SIZE**	*width,* *height*	Field width Field height
	TYPE	*fieldtype*	Type of field: TEXT (default) INVISIBLE PASSWORD (contents is echoed with '*')
	CLASS	*class*	Field class OPTIONAL (default) STATIC REQUIRED
	KEYS	*keys*	Accepted input key types: NUMERIC HEXADECIMAL ALPHANUMERIC
	ACCESS	*access*	Access rights of field WRITE (default) READ READWRITE
	OVERFLOW	*overflow*	Action on field overflow: TERMINATE (default) TRUNCATE OVERWRITE
	STYLE	*style*	Display attributes as a combination of the following, ORed together using the "|" operator: NORMAL (default) UNDER (single underline) INVERTED FLASHING
	HORIZONTAL	*justify*	Horizontal alignment of field contents LEFT (default) RIGHT CENTER
	FORMAT	*formatstring**	Application defined
	INITIALVALUE	*value**	Initial value
END			

7. C - Header file

```
/******************************************************************************
*                                                                             *
* xfsttu.h      XFS - definitions                                             *
*               for the Text Terminal Unit - services                         *
*                                                                             *
*               Version 2.00 (11/11/96)                                       *
*                                                                             *
******************************************************************************/

#ifndef __INC_XFSTTU__H
#define __INC_XFSTTU__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

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


/* values of WFSTTUCAPS.wClass */
#define     WFS_SERVICE_CLASS_TTU                   (7)
#define     WFS_SERVICE_CLASS_NAME_TTU              "TTU"
#define     WFS_SERVICE_CLASS_VERSION_TTU           (0x0002)

#define     TTU_SERVICE_OFFSET                      (WFS_SERVICE_CLASS_TTU * 100)

/* TTU Info Commands */
#define     WFS_INF_TTU_STATUS                      (TTU_SERVICE_OFFSET + 1)
#define     WFS_INF_TTU_CAPABILITIES                (TTU_SERVICE_OFFSET + 2)
#define     WFS_INF_TTU_FORM_LIST                   (TTU_SERVICE_OFFSET + 3)
#define     WFS_INF_TTU_QUERY_FORM                  (TTU_SERVICE_OFFSET + 4)
#define     WFS_INF_TTU_QUERY_FIELD                 (TTU_SERVICE_OFFSET + 5)

/* TTU Command Verbs */
#define     WFS_CMD_TTU_BEEP                        (TTU_SERVICE_OFFSET + 1)
#define     WFS_CMD_TTU_CLEARSCREEN                 (TTU_SERVICE_OFFSET + 2)
#define     WFS_CMD_TTU_DISPLIGHT                   (TTU_SERVICE_OFFSET + 3)
#define     WFS_CMD_TTU_SET_LED                     (TTU_SERVICE_OFFSET + 4)
#define     WFS_CMD_TTU_SET_RESOLUTION              (TTU_SERVICE_OFFSET + 5)
#define     WFS_CMD_TTU_DISPLAY_FORM                (TTU_SERVICE_OFFSET + 6)
#define     WFS_CMD_TTU_READ_FORM                   (TTU_SERVICE_OFFSET + 7)
#define     WFS_CMD_TTU_WRITE                       (TTU_SERVICE_OFFSET + 8)
#define     WFS_CMD_TTU_READ                        (TTU_SERVICE_OFFSET + 9)


/* XFS TTU Errors */

#define     WFS_ERR_TTU_FIELDERROR                  (-(TTU_SERVICE_OFFSET + 1))
#define     WFS_ERR_TTU_FIELDINVALID                (-(TTU_SERVICE_OFFSET + 2))
#define     WFS_ERR_TTU_FIELDNOTFOUND               (-(TTU_SERVICE_OFFSET + 3))
#define     WFS_ERR_TTU_FIELDSPECFAILURE            (-(TTU_SERVICE_OFFSET + 4))
#define     WFS_ERR_TTU_FORMINVALID                 (-(TTU_SERVICE_OFFSET + 5))
#define     WFS_ERR_TTU_FORMNOTFOUND                (-(TTU_SERVICE_OFFSET + 6))
#define     WFS_ERR_TTU_INVALIDLED                  (-(TTU_SERVICE_OFFSET + 7))
#define     WFS_ERR_TTU_KEYCANCELED                 (-(TTU_SERVICE_OFFSET + 8))
#define     WFS_ERR_TTU_MEDIAOVERFLOW               (-(TTU_SERVICE_OFFSET + 9))
#define     WFS_ERR_TTU_RESNOTSUPP                  (-(TTU_SERVICE_OFFSET + 10))


/* Values of WFSTTUSTATUS.fwDevice */
#define     WFS_TTU_DEVONLINE                       WFS_STAT_DEVONLINE
#define     WFS_TTU_DEVOFFLINE                      WFS_STAT_DEVOFFLINE
#define     WFS_TTU_DEVPOWEROFF                     WFS_STAT_DEVPOWEROFF
```

```
#define     WFS_TTU_DEVBUSY                     WFS_STAT_DEVBUSY
#define     WFS_TTU_DEVNODEVICE                 WFS_STAT_DEVNODEVICE
#define     WFS_TTU_DEVHWERROR                  WFS_STAT_DEVHWERROR
#define     WFS_TTU_DEVUSERERROR                WFS_STAT_DEVUSERERROR

/* Values of WFSTTUSTATUS.wKeyboard */
#define     WFS_TTU_KBDNA                       (0)
#define     WFS_TTU_KBDON                       (1)
#define     WFS_TTU_KBDOFF                      (2)

/* Values of WFSTTUSTATUS.wKeyLock */
#define     WFS_TTU_KBDLOCKNA                   (0)
#define     WFS_TTU_KBDLOCKON                   (1)
#define     WFS_TTU_KBDLOCKOFF                  (2)

#define     WFS_TTU_LEDS_MAX                    (8)

/* Values of WFSTTUSTATUS.fwLEDs */
#define     WFS_TTU_LEDNA                       (0x0000)
#define     WFS_TTU_LEDOFF                      (0x0001)
#define     WFS_TTU_LEDON                       (0x0002)
#define     WFS_TTU_LEDSLOWFLASH                (0x0004)
#define     WFS_TTU_LEDMEDIUMFLASH              (0x0008)
#define     WFS_TTU_LEDQUICKFLASH               (0x0010)
#define     WFS_TTU_LEDCONTINUOUS               (0x0080)

/* Values of WFSTTUCAPS.fwType */
#define     WFS_TTU_FIXED                       (0x0001)
#define     WFS_TTU_REMOVABLE                   (0x0002)

/* Values of WFSTTUCAPS.fwKeys */
#define     WFS_TTU_KEYNUMERIC                  (0x0001)
#define     WFS_TTU_KEYHEXADECIMAL              (0x0002)
#define     WFS_TTU_KEYALPHANUMERIC             (0x0004)

/* Values of WFSTTUFRMFIELD.fwType */
#define     WFS_TTU_FIELDTEXT                   (0)
#define     WFS_TTU_FIELDINVISIBLE              (1)
#define     WFS_TTU_FIELDPASSWORD               (2)

/* Values of WFSTTUFRMFIELD.fwClass */
#define     WFS_TTU_CLASSOPTIONAL               (0)
#define     WFS_TTU_CLASSSTATIC                 (1)
#define     WFS_TTU_CLASSREQUIRED               (2)

/* Values of WFSTTUFRMFIELD.fwAccess */
#define     WFS_TTU_ACCESSREAD                  (0x0001)
#define     WFS_TTU_ACCESSWRITE                 (0x0002)

/* Values of WFSTTUFRMFIELD.fwOverflow */
#define     WFS_TTU_OVFTERMINATE                (0)
#define     WFS_TTU_OVFTRUNCATE                 (1)
#define     WFS_TTU_OVFOVERWRITE                (2)

/* Values of WFSTTUWRITE.fwMode */
#define     WFS_TTU_POSRELATIVE                 (0)
#define     WFS_TTU_POSABSOLUTE                 (1)

/* Values of WFSTTUWRITE.fwTextAttr */
#define     WFS_TTU_TEXTUNDERLINE               (0x0001)
#define     WFS_TTU_TEXTINVERTED                (0x0002)
#define     WFS_TTU_TEXTFLASH                   (0x0004)

/* Values of WFSTTUFRMREAD.fwEchoMode */
#define     WFS_TTU_ECHOTEXT                    (0)
#define     WFS_TTU_ECHOINVISIBLE               (1)
#define     WFS_TTU_ECHOPASSWORD                (2)

#define     WFS_TTU_BEEPOFF                     (0x0001)
```

```
#define     WFS_TTU_BEEPKEYPRESS                (0x0002)
#define     WFS_TTU_BEEPEXCLAMATION             (0x0004)
#define     WFS_TTU_BEEPWARNING                 (0x0008)
#define     WFS_TTU_BEEPERROR                   (0x0010)
#define     WFS_TTU_BEEPCRITICAL                (0x0020)
#define     WFS_TTU_BEEPCONTINUOUS              (0x0080)

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

typedef struct _wfs_ttu_status
{
    WORD            fwDevice;
    WORD            wKeyboard;
    WORD            wKeylock;
    WORD            wLEDs[WFS_TTU_LEDS_MAX];
    WORD            wDisplaySizeX;
    WORD            wDisplaySizeY;
    LPSTR           lpszExtra;
} WFSTTUSTATUS, * LPWFSTTUSTATUS;

typedef struct _wfs_ttu_resolution
{
    WORD            wSizeX;
    WORD            wSizeY;
} WFSTTURESOLUTION, * LPWFSTTURESOLUTION;

typedef struct _wfs_ttu_caps
{
    WORD                  wClass;
    WORD                  fwType;
    LPWFSTTURESOLUTION *  lppResolutions;
    WORD                  wNumOfLEDs;
    BOOL                  bKeyLock;
    BOOL                  bDisplayLight;
    WORD                  fwKeys;
    BOOL                  bCursor;
    BOOL                  bForms;
    LPSTR                 lpszExtra;
} WFSTTUCAPS, * LPWFSTTUCAPS;

typedef struct _wfs_ttu_frm_header
{
    LPSTR           lpszFormName;
    WORD            wWidth;
    WORD            wHeight;
    WORD            wVersionMajor;
    WORD            wVersionMinor;
    LPSTR           lpszFields;
} WFSTTUFRMHEADER, * LPWFSTTUFRMHEADER;

typedef struct _wfs_ttu_query_field
{
    LPSTR           lpszFormName;
    LPSTR           lpszFieldName;
} WFSTTUQUERYFIELD, * LPWFSTTUQUERYFIELD;

typedef struct _wfs_ttu_frm_field
{
    LPSTR           lpszFieldName;
    WORD            fwType;
    WORD            fwClass;
    WORD            fwAccess;
    WORD            fwOverflow;
    LPSTR           lpszFormat;
} WFSTTUFRMFIELD, * LPWFSTTUFRMFIELD;

typedef struct _wfs_ttu_clear_screen
```

```
{
    WORD            wPositionX;
    WORD            wPositionY;
    WORD            wWidth;
    WORD            wHeight;
} WFSTTUCLEARSCREEN, * LPWFSTTUCLEARSCREEN;

typedef struct _wfs_ttu_disp_light
{
    BOOL            bMode;
} WFSTTUDISPLIGHT, * LPWFSTTUDISPLIGHT;

typedef struct _wfs_ttu_set_leds
{
    WORD            wLED;
    WORD            fwCommand;
} WFSTTUSETLEDS, * LPWFSTTUSETLEDS;

typedef struct _wfs_ttu_display_form
{
    LPSTR           lpszFormName;
    BOOL            bClearScreen;
    LPSTR           lpszFields;
} WFSTTUDISPLAYFORM, * LPWFSTTUDISPLAYFORM;

typedef struct _wfs_ttu_read_form
{
    LPSTR           lpszFormName;
    LPSTR           lpszFieldNames;
} WFSTTUREADFORM, * LPWFSTTUREADFORM;

typedef struct _wfs_ttu_write
{
    WORD            fwMode;
    WORD            wPosX;
    WORD            wPosY;
    WORD            fwTextAttr;
    LPSTR           lpsText;
} WFSTTUWRITE, * LPWFSTTUWRITE;

typedef struct _wfs_ttu_read
{
    WORD            wNumOfChars;
    WORD            fwMode;
    WORD            wPosX;
    WORD            wPosY;
    WORD            fwEchoMode;
    WORD            fwEchoAttr;
    WORD            wKeys;
    BOOL            bCursor;
    BOOL            bFlush;
    BOOL            bAutoEnd;
} WFSTTUREAD, * LPWFSTTUREAD;


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

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

#endif  /* __INC_XFSTTU__H */
```