



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

WORKSHOP AGREEMENT

CWA 14050-22

November 2000

ICS 35.200; 35.240.40

Extensions for Financial Services (XFS) interface specification -
Release 3.0 - Part 22: Text Terminal Unit Device Class Interface -
Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) -
Programmer's Reference

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

© 2000 CEN

All rights of exploitation in any form and by any means reserved world-wide for CEN National Members

Ref. No CWA 14050-22:2000 E

Table of Contents

Foreword	3
1. General	5
2. New Chapters.....	5
2.1 References	5
2.2 XFS form/media definition files in multi-vendor environments.....	5
3. New Info Commands.....	5
3.1 WFS_INF_TTU_KEY_DETAIL	5
4. Changes to existing Info Commands	7
4.1 WFS_INF_TTU_STATUS	7
4.2 WFS_INF_TTU_CAPABILITIES.....	9
4.3 WFS_INF_TTU_QUERY_FORM.....	10
4.4 WFS_INF_TTU_QUERY_FIELD	11
5. New Execute Commands.....	12
5.1 WFS_CMD_TTU_RESET	12
6. Changes to existing Execute Commands.....	13
6.1 WFS_CMD_TTU_CLEARSCREEN.....	13
6.2 WFS_CMD_TTU_WRITE_FORM (former WFS_CMD_TTU_DISPLAY_FORM).....	13
6.3 WFS_CMD_TTU_READ_FORM	14
6.4 WFS_CMD_TTU_WRITE	15
6.5 WFS_CMD_TTU_READ.....	16
7. New Events	19
7.1 WFS_EXEE_TTU_FIELDERROR	19
7.2 WFS_EXEE_TTU_FIELDWARNING.....	20
7.3 WFS_EXEE_TTU_KEY	20
8. Changes to existing Events.....	21
9. Changes to Form and Field Definitions	21
9.1 Definition Syntax	21
9.2 Field Definition	22
10. Changes to C - Header file.....	23

Foreword

This CWA is revision 3.0 of the XFS interface specification.

The move from an XFS 2.0 specification (CWA 13449) to a 3.0 specification has been prompted by a series of factors.

Initially, there has been a technical imperative to extend the scope of the existing specification of the XFS Manager to include new devices, such as the Card Embossing Unit.

Similarly, there has also been pressure, through implementation experience and the advance of the Microsoft technology, to extend the functionality and capabilities of the existing devices covered by the specification.

Finally, it is also clear that our customers and the market are asking for an update to a specification, which is now over 2 years old. Increasing market acceptance and the need to meet this demand is driving the Workshop towards this release.

The clear direction of the CEN/ISSS XFS Workshop, therefore, is the delivery of a new Release 3.0 specification based on a C API. It will be delivered with the promise of the protection of technical investment for existing applications and the design to safeguard future developments.

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

This CWA was formally approved by the XFS Workshop meeting on 2000-10-18. The specification is continuously reviewed and commented in the CEN/ISSS Workshop on XFS. It is therefore expected that an update of the specification will be published in due time as a CWA, superseding this revision 3.0.

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

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

Part 2: Service Classes Definition; Programmer's Reference

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

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

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

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

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

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

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

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

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

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

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

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

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

Part 16: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 17: Printer Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 18: Identification Card Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 19: Cash Dispenser Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 20: PIN Keypad Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 21: Depository Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 22: Text Terminal Unit Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 23: Sensors and Indicators Unit Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 24: Camera Device Class Interface - Migration from Version 2.0 (see CWA 13449) to Version 3.0 (this CWA) - Programmer's Reference

Part 25: Identification Card Device Class Interface - PC/SC Integration Guidelines

In addition to these Programmer's Reference specifications, the reader of this CWA is also referred to a complementary document, called Release Notes. The Release Notes contain clarifications and explanations on the CWA specifications, which are not requiring functional changes. The current version of the Release Notes is available online from <http://www.cenorm.be/isss/Workshop/XFS>.

The information in this document 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.

1. General

UNICODE support for field values, key stroke events during text entry and a few clarifications have been added. For all commands, the error code and event sections have been amended to explicitly mention that the generic values from the API document can also be returned.

2. New Chapters

2.1 References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer's Reference
Revision 3.0, October 18, 2000

2.2 XFS form/media definition files in multi-vendor environments

Although for most Service Providers directory location and extension of XFS form/media definition files are configurable through the registry, the capabilities of Service Providers and or actual hardware may vary. Therefore the following considerations should be taken into account when applications use XFS form definition files with the purpose of running in a multi-vendor environment:

- Physical display area dimensions may vary from one text terminal to another
- Just-in-time form loading may not be supported by all Service Providers, which makes it impossible to create dynamic form files just before displaying them (which in return means that only the display data of the forms can be changed, not the -layout data such as field positions)
- Some form/media definition keywords may not be supported due to limitations of the hardware or software

3. New Info Commands

3.1 WFS_INF_TTU_KEY_DETAIL

Description This command returns information about the Keys (buttons) supported by the device.

This command should be issued to determine which Keys are available.

Input Param None.

Output Param LPWFSTTUKEYDETAIL lpKeyDetail;

```
typedef struct _wfs_ttu_key_detail
{
    LPSTR          lp.szKeys;
    LPWSTR         lp.wszUNICODEKeys;
    LPWORD         lp.wCommandKeys;
} WFSSTTUKEYDETAIL, * LPWFSTTUKEYDETAIL;
```

lp.szKeys

String which holds the printable characters (numeric and alphanumeric keys) on the Text Terminal Unit, e.g. "0123456789ABCabcαβχ" if those text terminal input keys are present. This string is a NULL pointer if capability *fwCharSupport* equals WFS_TTU_UNICODE or if no keys of this type are present on the device.

lp.wszUNICODEKeys

String which holds the numeric and alphanumeric keys on the Text Terminal Unit like *lp.szKeys* but

in UNICODE format. This string is a NULL pointer if capability *fwCharSupport* equals WFS_TTU_ASCII or if no keys of this type are present on the device.

lpwCommandKeys

Array of command keys on the Text Terminal Unit. The array is terminated with a zero value. This array is a NULL pointer if no keys of this type are present on the device.

WFS_TTU_CK_ENTER
WFS_TTU_CK_CANCEL
WFS_TTU_CK_CLEAR
WFS_TTU_CK_BACKSPACE
WFS_TTU_CK_HELP
WFS_TTU_CK_00
WFS_TTU_CK_000
WFS_TTU_CK_ARROWUP
WFS_TTU_CK_ARROWDOWN
WFS_TTU_CK_ARROWLEFT
WFS_TTU_CK_ARROWRIGHT

The following values may be used as vendor dependent keys.

WFS_TTU_CK_OEM1
WFS_TTU_CK_OEM2
WFS_TTU_CK_OEM3
WFS_TTU_CK_OEM4
WFS_TTU_CK_OEM5
WFS_TTU_CK_OEM6
WFS_TTU_CK_OEM7
WFS_TTU_CK_OEM8
WFS_TTU_CK_OEM9
WFS_TTU_CK_OEM10
WFS_TTU_CK_OEM11
WFS_TTU_CK_OEM12

The following keys are used for Function Descriptor Keys.

WFS_TTU_CK_FDK01
WFS_TTU_CK_FDK02
WFS_TTU_CK_FDK03
WFS_TTU_CK_FDK04
WFS_TTU_CK_FDK05
WFS_TTU_CK_FDK06
WFS_TTU_CK_FDK07
WFS_TTU_CK_FDK08
WFS_TTU_CK_FDK09

WFS_TTU_CK_FDK10
WFS_TTU_CK_FDK11
WFS_TTU_CK_FDK12
WFS_TTU_CK_FDK13
WFS_TTU_CK_FDK14
WFS_TTU_CK_FDK15
WFS_TTU_CK_FDK16
WFS_TTU_CK_FDK17
WFS_TTU_CK_FDK18
WFS_TTU_CK_FDK19
WFS_TTU_CK_FDK20
WFS_TTU_CK_FDK21
WFS_TTU_CK_FDK22
WFS_TTU_CK_FDK23
WFS_TTU_CK_FDK24
WFS_TTU_CK_FDK25
WFS_TTU_CK_FDK26
WFS_TTU_CK_FDK27
WFS_TTU_CK_FDK28
WFS_TTU_CK_FDK29
WFS_TTU_CK_FDK30
WFS_TTU_CK_FDK31
WFS_TTU_CK_FDK32

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

Comments None.

4. Changes to existing Info Commands

4.1 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         lpzExtra;
} WFS_TTU_STATUS, * LPWFS_TTU_STATUS;
```

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 (i.e., powered on and operable).
WFS_TTU_DEVOFFLINE	The device is off-line (e.g., the operator has taken the device offline by turning a switch or pulling out the device).
WFS_TTU_DEVPOWEROFF	The device is powered off or physically not connected.
WFS_TTU_DEVBUSY	The device is busy and unable to process an execute command at this time.
WFS_TTU_DEVNODEVICE	There is no device intended to be there; e.g. this type of self service machine does not contain such a device or it is internally not configured.
WFS_TTU_DEVHWERROR	The device is inoperable due to a hardware error.
WFS_TTU_DEVUSERERROR	The device is inoperable because a person is preventing proper device operation.

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

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

4.2 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;
    WORD                fwCharSupport;
    LPSTR               lpSzExtra;
} WFSTTUCAPS, * LPWFSTTUCAPS;
```

(NOTE – This field has been replaced by the WFS_INF_TTU_KEY_DETAIL info command)

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.

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).

fwCharSupport

One or more flags specifying the Character Sets, in addition to single byte ASCII, supported by the service provider:

Value	Meaning
WFS_TTU_ASCII	ASCII is supported for XFS forms.
WFS_TTU_UNICODE	UNICODE is supported for XFS forms.

For *fwCharSupport*, a service provider can support ONLY ASCII forms or can support BOTH ASCII and UNICODE forms. A service provider can not support UNICODE forms without also supporting ASCII forms.

lpzExtra

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

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

4.3 WFS_INF_TTU_QUERY_FORM

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

Input Param LPSTR *lpzFormName* ;

lpzFormName

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

Output Param LPWFSTTUFRMHEADER *lpFrmHeader* ;

```
typedef struct _wfs_ttu_frm_header
{
    LPSTR        lpzFormName ;
    WORD        wWidth ;
    WORD        wHeight ;
    WORD        wVersionMajor ;
    WORD        wVersionMinor ;
    WORD        fwCharSupport ;
    LPSTR        lpzFields ;
} WFSSTTUFRMHEADER, * LPWFSTTUFRMHEADER ;
```

lpzFormName

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.

fwCharSupport

A single flag indicating whether the form is encoded in ASCII or UNICODE:

Value	Meaning
WFS_TTU_ASCII	XFS form is encoded in ASCII.
WFS_TTU_UNICODE	XFS form is encoded in UNICODE.

lpszFields

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

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

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

Comments None.

4.4 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;
} WFSSTTUQUERYFIELD, * 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 a **NULL pointer**, 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;
} WFSSTTUFRMFIELD, * 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.

lpzFormat

Format string as defined in the form for this field.

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

Value	Meaning
WFS_ERR_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.

5. New Execute Commands

5.1 WFS_CMD_TTU_RESET

Description Sends a service reset to the service provider. This command clears the screen and the keyboard buffer.

Input Param None

Output Param None.

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

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

Comments This command is used by an application control program to cause a device to reset itself to a known good condition.

6. Changes to existing Execute Commands

6.1 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;
} WFSSTTUCLEARSCREEN, * 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. This value must be positive.

wHeight
Specifies the height of the area to be cleared. This value must be positive.

Output Param None.

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

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

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

6.2 WFS_CMD_TTU_WRITE_FORM (former 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 lpWriteform;

```
typedef struct _wfs_ttu_write_form
{
    LPSTR        lpszFormName;
    BOOL        bClearScreen;
    LPSTR        lpszFields;
    LPWSTR        lpszUNICODEFields;
} WFSSTTUWRITEFORM, * LPWFSTTUWRITEFORM;
```

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 entire field string terminating with two null characters, e.g. Field1=123/0Field2=456/0/0. The <FieldValue> stands for a string containing all the printable characters (numeric and alphanumeric) to display on the text terminal unit key pad for this field.

lpszUNICODEFields

Pointer to a series of "<FieldName>=<FieldValue>" UNICODE strings, where each string is null-terminated with the entire field string terminating with two null characters, e.g. Field1=123/0Field2=456/0/0 (UNICODE). The <FieldValue> stands for a UNICODE string containing all the printable characters (numeric and alphanumeric) to display on the text terminal unit key pad for this field.

Note: The *lpszUNICODEFields* field should only be used if the form is encoded in UNICODE representation. This can be determined with the WFS_TTU_INF_QUERY_FORM command. The use of *lpszFields* and *lpszUNICODEFields* fields is mutually exclusive.

Output Param None.

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

Value	Meaning
WFS_ERR_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 <i>lpszFields</i> member is invalid.
WFS_ERR_TTU_CHARSETDATA	Character set(s) supported by service provider is inconsistent with use of <i>lpszFields</i> or <i>lpszUNICODEFields</i> fields.
WFS_ERR_TTU_FIELDERROR	An error occurred while processing a field, causing termination of the display request

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

Value	Meaning
WFS_EXEE_TTU_FIELDERROR	A fatal error occurred while processing a field.
WFS_EXEE_TTU_FIELDWARNING	A non-fatal error occurred while processing a field.

Comments None.

6.3 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;
} WFSSTTUREADFORM, * 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 a NULL pointer, then data is read from all input fields on the form.

Output Param LPWFSTTUREADFORMOUT lpReadFormOut;

```
typedef struct _wfs_ttu_read_form_out
{
    LPSTR    lpszFields;
    LPWSTR   lpszUNICODEFields;
} WFSSTTUREADFORMOUT, * LPWFSTTUREADFORMOUT;
```

lpzFields

Pointer to a series of "<FieldName>=<FieldValue>" strings, where each string is null-terminated with the final string terminating with two null characters, e.g. Field1=123/0Field2=456/0/0. The <FieldValue> stands for a string containing all the printable characters (numeric and alphanumeric) read from the text terminal unit key pad for this field. This parameter is a NULL pointer if the capability *fwCharSupport* equals WFS_TTU_UNICODE.

lpzUNICODEFields

Pointer to a series of "<FieldName>=<FieldValue>" UNICODE strings, where each string is null-terminated with the entire field string terminating with two null characters, e.g. Field1=123/0Field2=456/0/0 (UNICODE). The <FieldValue> stands for a UNICODE string containing all the printable characters (numeric and alphanumeric) read from the text terminal unit key pad for this field. This parameter is a NULL pointer if the capability *fwCharSupport* equals WFS_TTU_ASCII.

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

Value	Meaning
WFS_ERR_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 <i>lpzFieldNames</i> member is invalid.
WFS_ERR_TTU_KEYCANCELED	The read operation was terminated by pressing the <CANCEL>-key.
WFS_ERR_TTU_FIELDERROR	An error occurred while processing a field, causing termination of the read request.

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

Value	Meaning
WFS_EXEE_TTU_FIELDERROR	A fatal error occurred while processing a field.
WFS_EXEE_TTU_FIELDWARNING	A non-fatal error occurred while processing a field..

Comments None.

6.4 WFS_CMD_TTU_WRITE

Description This command displays the specified text on the display of the text terminal unit . The specified text may include the control characters CR (Carriage Return) and LF (Line Feed). The control characters can be included in the text as CR, or LF, or CR LF, or LF CR and all combinations will perform the function of relocating the cursor position to the left hand side of the display on the next line down. If the text will overwrite the display area then the display will scroll.

Input Param LPWFSTTUWRITE lpWrite;

```
typedef struct _wfs_ttu_write
{
    WORD          fwMode;
    SHORT         wPosX;
    SHORT         wPosY;
    WORD          fwTextAttr;
    LPSTR         lpsText;
    LPWSTR        lpsUNICODEText;
} WFS_TTUWRITE, * 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 <i>wPosX</i> and <i>wPosY</i> .

wPosX

If *fwMode* is set to WFS_TTU_POSABSOLUTE, this specifies the absolute horizontal position. If *fwMode* is set to WFS_TTU_POSRELATIVE this specifies a horizontal offset relative to the current cursor position as a zero (0) based value.

wPosY

If *fwMode* is set to WFS_TTU_POSABSOLUTE, this specifies the absolute vertical position. If *fwMode* is set to WFS_TTU_POSRELATIVE this specifies a vertical offset relative to the current cursor position as a zero (0) based value.

fwTextAttr

Specifies the text attributes used for displaying the text as a combination of the following flags. If none of the following attribute flags are selected then the text will be displayed as TEXTNORMAL.

Value	Meaning
WFS_TTU_TEXTUNDERLINE	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.

lpsUNICODEText

Specifies the UNICODE text that will be displayed.

Note: *lpsText* and *lpsUNICODEText* are mutually exclusive.

Output Param None.

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

Value	Meaning
WFS_ERR_TTU_CHARSETDATA	Character set(s) supported by service provider is inconsistent with use of <i>lpsText</i> or <i>lpsUNICODEText</i> fields.

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

Comments None.

6.5 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. During this command, pressing an active key results in a WFS_EXEE_TTU_KEY event containing the key details. On completion of the command (when the maximum number of keys have been pressed or a terminator key is pressed), the entered string, as interpreted by the service provider, is returned. The service provider takes command keys into account when interpreting the data.

Input Param

```
LPWFSTTUREAD lpRead;

typedef struct _wfs_ttu_read
{
    WORD        wNumOfChars;
    WORD        fwMode;
    SHORT       wPosX;
    SHORT       wPosY;
    WORD        fwEchoMode;
    WORD        fwEchoAttr;
    BOOL        bCursor;
    BOOL        bFlush;
    BOOL        bAutoEnd;
    LPSTR       lpzActiveKeys;
    LPWSTR      lpwszActiveUNICODEKeys;
    LPWORD      lpwActiveCommandKeys;
    LPWORD      lpwTerminateCommandKeys;
} WFSSTTUREAD, * LPWFSTTUREAD;
```

wNumOfChars

Specifies the number of printable characters (numeric and alphanumeric keys) that will be read from the text terminal unit key pad. All command keys like WFS_TTU_CK_ENTER, WFS_TTU_CK_FDK01 will not be counted.

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 <i>wPosX</i> and <i>wPosY</i> .

wPosX

If *fwMode* is set to WFS_TTU_POSABSOLUTE, this specifies the absolute horizontal position. If *fwMode* is set to WFS_TTU_POSRELATIVE this specifies a horizontal offset relative to the current cursor position as a zero (0) based value.

wPosY

If *fwMode* is set to WFS_TTU_POSABSOLUTE, this specifies the absolute vertical position. If *fwMode* is set to WFS_TTU_POSRELATIVE this specifies a vertical offset relative to the current cursor position as a zero (0) based value.

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. If none of the following attribute flags are selected then the text will be displayed as TEXTNORMAL.

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

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 printable characters as specified with *wNumOfChars* is entered.

lpSzActiveKeys

String which specifies the numeric and alphanumeric keys on the Text Terminal Unit, e.g. "12ABab", to be active during the execution of the command. Devices having a shift key interpret this parameter differently from those that do not have a shift key. For devices having a shift key, specifying only the upper case of a particular letter enables both upper and lower case of that key, but the device converts lower case letters to upper case in the output parameter. To enable both upper and lower case keys, and have both upper and lower case letters returned, specify both the upper and lower case of the letter (e.g. "12AaBb"). For devices not having a shift key, specifying either the upper case only (e.g. "12AB"), or specifying both the upper and lower case of a particular letter (e.g. "12AaBb"), enables that key and causes the device to return the upper case of the letter in the output parameter. For both types of device, specifying only lower case letters (e.g. "12ab") produces a key invalid error. This parameter is a NULL pointer if capability *fwCharSupport* equals *WFS_TTU_UNICODE* or if no keys of this type are active keys.

lpwSzActiveUNICODEKeys

String which specifies the numeric and alphanumeric keys on the Text Terminal Unit, e.g. "12ABab" (UNICODE), to be active during the execution of the command. Devices having a shift key interpret this parameter differently from those that do not have a shift key. For devices having a shift key, specifying only the upper case of a particular letter enables both upper and lower case of that key, but the device converts lower case letters to upper case in the output parameter. To enable both upper and lower case keys, and have both upper and lower case letters returned, specify both the upper and lower case of the letter (e.g. "12AaBb"). For devices not having a shift key, specifying either the upper case only (e.g. "12AB"), or specifying both the upper and lower case of a particular letter (e.g. "12AaBb"), enables that key and causes the device to return the upper case of the letter in the output parameter. For both types of device, specifying only lower case letters (e.g. "12ab") produces a key invalid error. This parameter is a NULL pointer if capability *fwCharSupport* equals *WFS_TTU_ASCII* or if no keys of this type are active keys.

lpwActiveCommandKeys

Array specifying the command keys which are active during the execution of the command. The array is terminated with a zero value and this array is a NULL pointer if no keys of this type are active keys.

lpwTerminateCommandKeys

Array specifying the command keys which must terminate the execution of the command. The array is terminated with a zero value and this array is a NULL pointer if no keys of this type are terminate keys.

Output Param LPWFSTTUREADIN lpReadIn;

```
typedef struct _wfs_ttu_read_in
{
    LPSTR lpSzInput;
    LPWSTR lpwSzUNICODEInput;
} WFSSTTUREADIN, * LPWFSTTUREADIN;
```

lpSzInput

Specifies a zero terminated string containing all the printable characters (numeric and alphanumeric) read from the text terminal unit key pad. This parameter is a NULL pointer if the capability *fwCharSupport* equals *WFS_TTU_UNICODE*.

lpzUNICODEInput

Specifies a zero terminated string containing all the printable characters (numeric and alphanumeric) read from the text terminal unit key pad. This parameter is a NULL pointer if the capability *fwCharSupport* equals *WFS_TTU_ASCII*.

Note: The following keys are not printable and will not be returned in the output parameter *lpzInput* or *lpzUNICODEInput*, but they may effect the buffer if active:

WFS_TTU_CK_CLEAR	Will clear the buffer. The number of printable characters pressed will be set to zero.
WFS_TTU_CK_BACKSPACE	Will cause the last printable character in the buffer to be removed. The number of printable characters pressed will be reduced by one, unless the number of printable characters pressed was zero.
WFS_TTU_CK_00	Will add a double zero '00' string to the buffer. The number of printable characters pressed will be increased by two.
WFS_TTU_CK_000	will add a triple zero '000' string to the buffer. The number of printable characters pressed will be increased by three.

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

Value	Meaning
WFS_ERR_TTU_KEYINVALID	At least one of the specified keys is invalid.
WFS_ERR_TTU_KEYNOTSUPPORTED	At least one of the specified keys is not supported by the service provider.
WFS_ERR_TTU_NOACTIVEKEYS	There are no active keys specified.

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

Value	Meaning
WFS_EXEE_TTU_KEY	An active key on the Text Terminal Unit has been pressed. Note: A command key press will not result in a character being displayed.

Comments None.

7. New Events

7.1 WFS_EXEE_TTU_FIELDERROR

Description This event specifies that a fatal error has occurred while processing a field.

Event Param LPWFSTTUFIELDFAIL lpFieldFail;

```
typedef struct _wfs_ttu_field_failure
{
    LPSTR      lpzFormName;
    LPSTR      lpzFieldName;
    WORD       wFailure;
} WFSSTTUFIELDFAIL, * LPWFSTTUFIELDFAIL;
```

lpzFormName

Points to the null-terminated form name.

lpzFieldName

Points to the null-terminated field name.

wFailure

Specifies the type of failure and can be one of the following:

Value	Meaning
WFS_TTU_FIELDREQUIRED	The specified field <i>must</i> be supplied by the application.
WFS_TTU_FIELDSTATICOVWR	The specified field is static and thus <i>cannot</i> be overwritten by the application.
WFS_TTU_FIELDOVERFLOW	The value supplied for the specified fields is too long.
WFS_TTU_FIELDNOTFOUND	The specified field does not exist.
WFS_TTU_FIELDNOTREAD	The specified field is not an input field.
WFS_TTU_FIELDNOTWRITE	An attempt was made to write to an input field.
WFS_TTU_FIELDTYPENOTSUPPORTED	The form field type is not supported with device.
WFS_TTU_CHARSETFORM	Service provider does not support character set specified in form.

Comments None.

7.2 WFS_EXEE_TTU_FIELDWARNING

Description This event is used to specify that a non-fatal error has occurred while processing a field.

Event Param LPWFSTTUFIELDFAIL lpFieldFail;
as defined in the section describing WFS_EXEE_TTU_FIELDERROR.

Comments None.

7.3 WFS_EXEE_TTU_KEY

Description This event specifies that any active key has been pressed at the TTU during the WFS_CMD_TTU_READ command. In addition to giving the application more details about individual key presses this information may also be used if the device has no internal display unit and the application has to manage the display of the entered digits.

Event Param LPWFSTTUKEY lpKey;

```
typedef struct _wfs_ttu_key
{
    CHAR        cKey;
    WORD        wUNICODEKey;
    WORD        wCommandKey;
} WFSSTTUKEY, * LPWFSTTUKEY;
```

cKey

On a numeric or alphanumeric key press this parameter holds the value of the key pressed. This value is WFS_TTU_NOKEY if no numeric or alphanumeric key was pressed or if capability *fwCharSupport* equals WFS_TTU_UNICODE.

wUNICODEKey

On a numeric or alphanumeric key press this parameter holds the value of the key pressed in UNICODE format. This value is WFS_TTU_NOKEY if no numeric or alphanumeric key was pressed or if capability *fwCharSupport* equals WFS_TTU_ASCII.

wCommandKey

On a Command key press this parameter holds the value of the Command key pressed, e.g. WFS_TTU_CK_ENTER. This value is WFS_TTU_NOKEY when no command key was pressed.

Note: Only one of the parameters *cKey*, *wUNICODEKey*, *wCommandKey* can have the value of a valid key, the others must be set to WFS_TTU_NOKEY.

Comments None.

8. Changes to existing Events

No changes

9. Changes to 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.

9.1 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.
- The order of attributes within a form is not mandatory; the attributes may be defined in any order.
- All forms can be represented using either ISO 646 (ANSI) or UNICODE character encoding. If the UNICODE representation is used then all Names and Strings are restricted to an internal representation of ISO 646 (ANSI) characters. Only the INITIALVALUE and FORMAT keyword values can have double byte values outside of the ISO 646 (ANSI) character set.
- If forms character encoding is UNICODE then, consistent with the UNICODE standard, the file prefix must be in little endian (xFFFE) or big endian (xFEFF) notation, such that UNICODE encoding is recognized.

9.2 Field Definition ¹

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

¹ Attributes are not required in any mandatory order within a Field Definition.

10. Changes to C - Header file

```

/*****
*
* xfsttu.h      XFS - definitions
*              for the Text Terminal Unit - services
*
*              Version 3.00 (10/18/00)
*
*****/

#ifndef __INC_XFSTTU__H
#define __INC_XFSTTU__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsttu.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 (0x0003)

#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)
#define WFS_INF_TTU_KEY_DETAIL (TTU_SERVICE_OFFSET + 6)

/* 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_WRITE_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)
#define WFS_CMD_TTU_RESET (TTU_SERVICE_OFFSET + 10)

/* TTU Messages */
#define WFS_EXEE_TTU_FIELDERROR (TTU_SERVICE_OFFSET + 1)
#define WFS_EXEE_TTU_FIELDWARNING (TTU_SERVICE_OFFSET + 2)
#define WFS_EXEE_TTU_KEY (TTU_SERVICE_OFFSET + 3)

/* Values of WFSTTUSTATUS.fwDevice */
#define WFS_TTU_DEVONLINE WFS_STAT_DEVONLINE
#define WFS_TTU_DEVOFFLINE WFS_STAT_DEVOFFLINE
#define WFS_TTU_DEVPPOWEROFF WFS_STAT_DEVPPOWEROFF
#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 (0x0002)
#define      WFS_TTU_LEDMEDIUMFLASH (0x0004)
#define      WFS_TTU_LEDQUICKFLASH (0x0008)
#define      WFS_TTU_LEDCONTINUOUS (0x0080)

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

/* Values of WFSTTUCAPS.fwCharSupport, WFSTTUWRITE.fwCharSupport */
#define      WFS_TTU_ASCII        (0x0001)
#define      WFS_TTU_UNICODE      (0x0002)

/* 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)

/* values of WFSTTUFIELDFAIL.wFailure */
```



```
#define WFS_TTU_FIELDREQUIRED (0)
#define WFS_TTU_FIELDSTATICOVWR (1)
#define WFS_TTU_FIELDOVERFLOW (2)
#define WFS_TTU_FIELDNOTFOUND (3)
#define WFS_TTU_FIELDNOTREAD (4)
#define WFS_TTU_FIELDNOTWRITE (5)
#define WFS_TTU_FIELDTYPENOTSUPPORTED (6)
#define WFS_TTU_CHARSETFORM (7)
```

```
/* values of WFS_TTU_KEYDETAIL.lpwCommandKeys */
```

```
#define WFS_TTU_NOKEY (0)
#define WFS_TTU_CK_ENTER (1)
#define WFS_TTU_CK_CANCEL (2)
#define WFS_TTU_CK_CLEAR (3)
#define WFS_TTU_CK_BACKSPACE (4)
#define WFS_TTU_CK_HELP (5)
#define WFS_TTU_CK_00 (6)
#define WFS_TTU_CK_000 (7)
#define WFS_TTU_CK_ARROWUP (8)
#define WFS_TTU_CK_ARROWDOWN (9)
#define WFS_TTU_CK_ARROWLEFT (10)
#define WFS_TTU_CK_ARROWRIGHT (11)
#define WFS_TTU_CK_OEM1 (12)
#define WFS_TTU_CK_OEM2 (13)
#define WFS_TTU_CK_OEM3 (14)
#define WFS_TTU_CK_OEM4 (15)
#define WFS_TTU_CK_OEM5 (16)
#define WFS_TTU_CK_OEM6 (17)
#define WFS_TTU_CK_OEM7 (18)
#define WFS_TTU_CK_OEM8 (19)
#define WFS_TTU_CK_OEM9 (20)
#define WFS_TTU_CK_OEM10 (21)
#define WFS_TTU_CK_OEM11 (22)
#define WFS_TTU_CK_OEM12 (23)
#define WFS_TTU_CK_FDK01 (24)
#define WFS_TTU_CK_FDK02 (25)
#define WFS_TTU_CK_FDK03 (26)
#define WFS_TTU_CK_FDK04 (27)
#define WFS_TTU_CK_FDK05 (28)
#define WFS_TTU_CK_FDK06 (29)
#define WFS_TTU_CK_FDK07 (30)
#define WFS_TTU_CK_FDK08 (31)
#define WFS_TTU_CK_FDK09 (32)
#define WFS_TTU_CK_FDK10 (33)
#define WFS_TTU_CK_FDK11 (34)
#define WFS_TTU_CK_FDK12 (35)
#define WFS_TTU_CK_FDK13 (36)
#define WFS_TTU_CK_FDK14 (37)
#define WFS_TTU_CK_FDK15 (38)
#define WFS_TTU_CK_FDK16 (39)
#define WFS_TTU_CK_FDK17 (40)
#define WFS_TTU_CK_FDK18 (41)
#define WFS_TTU_CK_FDK19 (42)
#define WFS_TTU_CK_FDK20 (43)
#define WFS_TTU_CK_FDK21 (44)
#define WFS_TTU_CK_FDK22 (45)
#define WFS_TTU_CK_FDK23 (46)
#define WFS_TTU_CK_FDK24 (47)
#define WFS_TTU_CK_FDK25 (48)
#define WFS_TTU_CK_FDK26 (49)
#define WFS_TTU_CK_FDK27 (50)
#define WFS_TTU_CK_FDK28 (51)
#define WFS_TTU_CK_FDK29 (52)
#define WFS_TTU_CK_FDK30 (53)
#define WFS_TTU_CK_FDK31 (54)
#define WFS_TTU_CK_FDK32 (55)
```

```
/* 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))
#define WFS_ERR_TTU_CHARSETDATA (- (TTU_SERVICE_OFFSET + 11))
#define WFS_ERR_TTU_KEYINVALID (- (TTU_SERVICE_OFFSET + 12))
#define WFS_ERR_TTU_KEYNOTSUPPORTED (- (TTU_SERVICE_OFFSET + 13))
#define WFS_ERR_TTU_NOACTIVEKEYS (- (TTU_SERVICE_OFFSET + 14))

/*=====*/
/* TTU Info Command Structures */
/*=====*/

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;
    BOOL bCursor;
    BOOL bForms;
    WORD fwCharSupport;
    LPSTR lpszExtra;
} WFSTTUCAPS, * LPWFSTTUCAPS;

typedef struct _wfs_ttu_frm_header
{
    LPSTR lpszFormName;
    WORD wWidth;
    WORD wHeight;
    WORD wVersionMajor;
    WORD wVersionMinor;
    WORD fwCharSupport;
    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_key_detail
{
    LPSTR          lpszKeys;
    LPWSTR         lpwUnicodeKeys;
    LPWORD         lpwCommandKeys;
} WFSTTUKEYDETAIL, * LPWFSTTUKEYDETAIL;

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_write_form
{
    LPSTR          lpszFormName;
    BOOL           bClearScreen;
    LPSTR          lpszFields;
    LPWSTR         lpwUnicodeFields;
} WFSTTUWRITEFORM, * LPWFSTTUWRITEFORM;

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

typedef struct _wfs_ttu_read_form_out
{
    LPSTR          lpszFields;
    LPWSTR         lpwUnicodeFields;
} WFSTTUREADFORMOUT, * LPWFSTTUREADFORMOUT;

typedef struct _wfs_ttu_write
{
    WORD           fwMode;
    SHORT          wPosX;
    SHORT          wPosY;
    WORD           fwTextAttr;
    LPSTR          lpszText;
    LPWSTR         lpwUnicodeText;
} WFSTTUWRITE, * LPWFSTTUWRITE;

typedef struct _wfs_ttu_read
{
    WORD           wNumOfChars;

```

```
WORD          fwMode;
SHORT         wPosX;
SHORT         wPosY;
WORD          fwEchoMode;
WORD          fwEchoAttr;
BOOL         bCursor;
BOOL         bFlush;
BOOL         bAutoEnd;
LPSTR        lpszActiveKeys;
LPWSTR       lpwActiveUNICODEKeys;
LPWORD       lpwActiveCommandKeys;
LPWORD       lpwTerminateCommandKeys;
} WFSTTUREAD, * LPWFSTTUREAD;

typedef struct _wfs_ttu_read_in
{
    LPSTR        lpszInput;
    LPWSTR       lpszUNICODEInput;
} WFSTTUREADIN, * LPWFSTTUREADIN;

/*=====*/
/* TTU Message Structures */
/*=====*/

typedef struct _wfs_ttu_field_failure
{
    LPSTR        lpszFormName;
    LPSTR        lpszFieldName;
    WORD         wFailure;
} WFSTTUFIELDFAIL, * LPWFSTTUFIELDFAIL;

typedef struct _wfs_ttu_key
{
    CHAR         cKey;
    WORD         wUNICODEKey;
    WORD         wCommandKey;
} WFSTTUKEY, * LPWFSTTUKEY;

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

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

#endif /* __INC_XFSTTU__H */
```