CEN

WORKSHOP

AGREEMENT

CWA 16374-9

December 2011

ICS 35.240.40

English version

Extensions for Financial Services (XFS) interface specification Release 3.20 - Part 9: Text Terminal Unit Device Class Interface Programmer's Reference

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

Management Centre: Avenue Marnix 17, B-1000 Brussels

Table of Contents

Foreword

This CWA is revision 3.20 of the XFS interface specification.

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties on 2011-06-29, the constitution of which was supported by CEN following the public call for participation made on 1998-06-24. 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 3.20.

A list of the individuals and organizations which supported the technical consensus represented by the CEN Workshop Agreement is available to purchasers from the CEN-CENELEC Management Centre. These organizations were drawn from the banking sector. The CEN/ISSS XFS Workshop gathered suppliers as well as banks and other financial service companies.

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

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

Part 2: Service Classes Definition - Programmer's Reference

Part 3: Printer and Scanning Device Class Interface Programmer's Reference

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

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

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

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

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

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

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

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

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

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

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

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

Part 16: Card Dispenser Device Class Interface - Programmer's Reference

Part 17: Barcode Reader Device Class Interface - Programmer's Reference

Part 18: Item Processing Module Device Class Interface- Programmer's Reference

Parts 19 - 28: Reserved for future use.

Parts 29 through 47 constitute an optional addendum to this CWA. They define the integration between the SNMP standard and the set of status and statistical information exported by the Service Providers.

Part 29: XFS MIB Architecture and SNMP Extensions

Part 30: XFS MIB Device Specific Definitions - Printer Device Class

Part 31: XFS MIB Device Specific Definitions - Identification Card Device Class

Part 32: XFS MIB Device Specific Definitions - Cash Dispenser Device Class

Part 33: XFS MIB Device Specific Definitions - PIN Keypad Device Class

Part 34: XFS MIB Device Specific Definitions - Check Reader/Scanner Device Class

Part 35: XFS MIB Device Specific Definitions - Depository Device Class

Part 36: XFS MIB Device Specific Definitions - Text Terminal Unit Device Class

Part 37: XFS MIB Device Specific Definitions - Sensors and Indicators Unit Device Class

Part 38: XFS MIB Device Specific Definitions - Camera Device Class

Part 39: XFS MIB Device Specific Definitions - Alarm Device Device Class

Part 40: XFS MIB Device Specific Definitions - Card Embossing Unit Class

Part 41: XFS MIB Device Specific Definitions - Cash-In Module Device Class

Part 42: Reserved for future use.

Part 43: XFS MIB Device Specific Definitions - Vendor Dependent Mode Class

Part 44: XFS MIB Application Management

Part 45: XFS MIB Device Specific Definitions - Card Dispenser Device Class

Part 46: XFS MIB Device Specific Definitions - Barcode Reader Device Class

Part 47: XFS MIB Device Specific Definitions - Item Processing Module Device Class

Parts 48 - 60 are reserved for future use.

Part 61: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 62: Printer and Scanning Device Class Interface - Migration from Version 3.10 (CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 63: Identification Card Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 65: PIN Keypad Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 67: Depository Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 71: Camera Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 72: Alarm Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.10 (CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 74: Cash-In Module Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 75: Card Dispenser Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 76: Barcode Reader Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

Part 77: Item Processing Module Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference

In addition to these Programmer's Reference specifications, the reader of this CWA is also referred to a

complementary document, called Release Notes. The Release Notes contain clarifications and explanations on the CWA specifications, which are not requiring functional changes. The current version of the Release Notes is available online from http://www.cen.eu/cen/pages/default.aspx.

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.

The formal process followed by the Workshop in the development of the CEN Workshop Agreement has been endorsed by the National Members of CEN but neither the National Members of CEN nor the CEN-CENELEC Management Centre can be held accountable for the technical content of the CEN Workshop Agreement or possible conflict with standards or legislation. This CEN Workshop Agreement can in no way be held as being an official standard developed by CEN and its members.

The final review/endorsement round for this CWA was started on 2011-06-23 and was successfully closed on 2011-07-23.The final text of this CWA was submitted to CEN for publication on 2011-08-26.

This CEN Workshop Agreement is publicly available as a reference document from the National Members of CEN: Austria, Belgium, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and the United Kingdom.

Comments or suggestions from the users of the CEN Workshop Agreement are welcome and should be addressed to the CEN-CENELEC Management Centre.

Revision History:

3.00	October 18, 2000	Initial release.
3.10	November 29, 2007	For a description of changes from version 3.00 to version 3.10 see the TTU 3.10 Migration document.
3.20	March 2nd, 2011	For a description of changes from version 3.10 to version 3.20 see the TTU 3.20 Migration document.

1. Introduction

1.1 Background to Release 3.20

The CEN/ISSS XFS Workshop aims to promote a clear and unambiguous specification defining a multi-vendor software interface to financial peripheral devices. The XFS (eXtensions for Financial Services) specifications are developed within the CEN/ISSS (European Committee for Standardization/Information Society Standardization System) Workshop environment. CEN/ISSS Workshops aim to arrive at a European consensus on an issue that can be published as a CEN Workshop Agreement (CWA).

The CEN/ISSS XFS Workshop encourages the participation of both banks and vendors in the deliberations required to create an industry standard. The CEN/ISSS XFS Workshop achieves its goals by focused sub-groups working electronically and meeting quarterly.

Release 3.20 of the XFS specification is based on a C API and is delivered with the continued promise for the protection of technical investment for existing applications. This release of the specification extends the functionality and capabilities of the existing devices covered by the specification, but does not include any new device classes. Notable major enhancements include Mixed Media processing to allow mixed cash and check accepting, as well as the addition of new commands to the CIM, PTR and IDC to allow better support of the Japanese marketplace.

1.2 XFS Service-Specific Programming

The service classes are defined by their service-specific commands and the associated data structures, error codes, messages, etc. These commands are used to request functions that are specific to one or more classes of Service Providers, but not all of them, and therefore are not included in the common API for basic or administration functions.

When a service-specific command is common among two or more classes of Service Providers, the syntax of the command is as similar as possible across all services, since a major objective of the XFS is to standardize function codes and structures for the broadest variety of services. For example, using the **WFSExecute** function, the commands to read data from various services are as similar as possible to each other in their syntax and data structures.

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

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

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

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

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

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

2. Text Terminal Units

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

All position indexes are zero based, where column zero, row zero is the top-leftmost position.

If the device has no shift key, the WFS_CMD_TTU_READ_FORM and WFS_CMD_TTU_READ commands will return only upper case letters. If the device has a shift key, these commands return upper and lower case letters as governed by the user's use of the shift key.

3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer's Reference Revision 3.20

4. 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                lpszExtra;
     WORD                 wDevicePosition;
     USHORT               usPowerSaveRecoveryTime;
     LPWFSTTULEDEX        lpLEDEx;
     WORD                 wAntiFraudModule;
     } 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 online (i.e. powered on and operable).
WFS_TTU_DEVOFFLINE	The device is offline (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.
WFS_TTU_DEVFRAUDATTEMPT	The device is present but is inoperable because it has detected a fraud attempt.
WFS_TTU_DEVPOTENTIALFRAUD	The device has detected a potential fraud attempt and is capable of remaining in service. In this case the application should make the decision as to whether to take the device offline.

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 LED index is WFS_TTU_LEDS_MAX -1. The number of available LEDs can be retrieved with the WFS_INF_TTU_CAPABILITIES info command. This field is only provided for backwards compatibility; the *lpLEDEx* parameter should instead be used to retrieve the LED status. 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
Pointer to a list of vendor-specific, or any other extended, information. The information is returned as a series of *"key=value"* strings so that it is easily extensible by Service Providers. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.

wDevicePosition
Specifies the device position. The device position value is independent of the *fwDevice* value, e.g. when the device position is reported as WFS_TTU_DEVICENOTINPOSITION, *fwDevice* can have any of the values defined above (including WFS_TTU_DEVONLINE or WFS_TTU_DEVOFFLINE). This value is one of the following values:

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

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

lpLEDEx
Pointer to a WFSTTULEDEX structure that specifies the states of the LEDs. If there is no LED available this will be a NULL pointer.

```
typedef struct _wfs_ttu_led_ex
     {
     USHORT                    usNumOfLEDs;
     LPDWORD                   lpdwLEDs;
     } WFSTTULEDEX, *LPWFSTTULEDEX;
```

usNumOfLEDs
This value specifies the number of LEDs, i.e. the size of the array returned in *lpdwLEDs*.

lpdwLEDs
Pointer to a DWORD array that specifies the state of each LED. Specifies the state of the LED as WFS_TTU_LEDNA, WFS_TTU_LEDOFF or a combination of the following flags consisting of one type B, and optionally one type C.

Value	Meaning	Type
WFS_TTU_LEDNA	The status is not available.	A
WFS_TTU_LEDOFF	The LED is turned off.	A
WFS_TTU_LEDSLOWFLASH	The LED is blinking slowly.	B
WFS_TTU_LEDMEDIUMFLASH	The LED is blinking medium frequency.	B
WFS_TTU_LEDQUICKFLASH	The LED is blinking quickly.	B
WFS_TTU_LEDCONTINUOUS	The LED is turned on continuous (steady).	B
WFS_TTU_LEDRED	The LED is red.	C
WFS_TTU_LEDGREEN	The LED is green.	C
WFS_TTU_LEDYELLOW	The LED is yellow.	C
WFS_TTU_LEDBLUE	The LED is blue.	C
WFS_TTU_LEDCYAN	The LED is cyan.	C
WFS_TTU_LEDMAGENTA	The LED is magenta.	C
WFS_TTU_LEDWHITE	The LED is white.	C

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

Value	Meaning
WFS_TTU_AFMNOTSUPP	No anti-fraud module is available.
WFS_TTU_AFMOK	Anti-fraud module is in a good state and no foreign device is detected.
WFS_TTU_AFMINOP	Anti-fraud module is inoperable.
WFS_TTU_AFMDEVICEDETECTED	Anti-fraud module detected the presence of a foreign device.
WFS_TTU_AFMUNKNOWN	The state of the anti-fraud module cannot be determined.

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

Comments Applications which require or expect specific information to be present in the *lpszExtra* parameter may not be device or vendor-independent.
In the case where communications with the device has been lost, the *fwDevice* field will report WFS_TTU_DEVPOWEROFF when the device has been removed or WFS_TTU_DEVHWERROR if the communications are unexpectedly lost. All other fields should contain a value based on the following rules and priority:

1. Report the value as unknown.
2. Report the value as a general h/w error.
3. Report the value as the last known value.

4.2 WFS_INF_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;
     BOOL                  bKeyLock;
     BOOL                  bDisplayLight;
     BOOL                  bCursor;
     BOOL                  bForms;
     WORD                  fwCharSupport;
     LPSTR                 lpszExtra;
     BOOL                  bPowerSaveControl;
     LPWFSTTULEDEX         lpLEDEx;
     BOOL                  bAntiFraudModule;
     } WFSTTUCAPS, *LPWFSTTUCAPS;
```

wClass
Specifies the logical service class as 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). The resolution indicated in the first position is the default resolution and the device will be placed in this resolution when the Service Provider is initialized or reset through the WFS_CMD_TTU_RESET command.

wNumOfLEDs
Specifies the number of LEDs available in this text terminal unit. This field is only provided for backwards compatibility; the *lpLEDEx* parameter should instead be used to retrieve the LED capabilities.

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 that can be switched ON and OFF with the WFS_CMD_TTU_DISPLIGHT command. 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.

lpszExtra
Pointer to a list of vendor-specific, or any other extended, information. The information is returned as a series of *"key=value"* strings so that it is easily extensible by Service Providers. Each string is null-terminated, with the final string terminating with two null characters. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.

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

lpLEDEx
Pointer to a WFSTTULEDEX structure that specifies the capabilities of the LEDs. If there is no LED available this will be a NULL pointer.

```
typedef struct _wfs_ttu_ledex
     {
     USHORT                    usNumOfLEDs;
     LPDWORD                   lpdwLEDs;
     } WFSTTULEDEX, *LPWFSTTULEDEX;
```

usNumOfLEDs
This value specifies the number of available LEDs, i.e. the size of the array returned in *lpdwLEDs*.

lpdwLEDs
Pointer to a DWORD array that specifies which LEDs are available.

The elements of this array are specified as a combination of the following flags and indicate all of the possible flash rates (type B) and colors (type C) that the LED is capable of handling. If the LED only supports one color then no value of type C is returned.

Value	Meaning	Type
WFS_TTU_LEDOFF	The LED can be off.	A
WFS_TTU_LEDSLOWFLASH	The LED can blink slowly.	B
WFS_TTU_LEDMEDIUMFLASH	The LED can blink medium frequency.	B
WFS_TTU_LEDQUICKFLASH	The LED can blink quickly.	B
WFS_TTU_LEDCONTINUOUS	The LED can be continuous (steady).	B
WFS_TTU_LEDRED	The LED can be red.	C
WFS_TTU_LEDGREEN	The LED can be green.	C
WFS_TTU_LEDYELLOW	The LED can be yellow.	C
WFS_TTU_LEDBLUE	The LED can be blue.	C
WFS_TTU_LEDCYAN	The LED can be cyan.	C
WFS_TTU_LEDMAGENTA	The LED can be magenta.	C
WFS_TTU_LEDWHITE	The LED can be white.	C

bAntiFraudModule
Specifies whether the anti-fraud module is available. This can either be TRUE if available or FALSE if not available.

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.

4.3 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	Only the generic error codes defined in [Ref. 1] can be generated by this command.
Comments	None.

4.4 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;
     WORD                fwCharSupport;
     LPSTR               lpszFields;
     WORD                wLanguageID;
     } 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. If the version is not specified in the form then zero is returned.

wVersionMinor
Specifies the minor version. If the version is not specified in the form then zero is returned.

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.

wLanguageID
Specifies the language identifier for the form.

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.

4.5 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 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;
     WORD                 wLanguageID;
     } 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.

lpszFormat
Format string as defined in the form for this field.

wLanguageID
Specifies the language identifier for the 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.

4.6 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               lpszKeys;
     LPWSTR              lpwszUNICODEKeys;
     LPWORD              lpwCommandKeys;
     } WFSTTUKEYDETAIL, *LPWFSTTUKEYDETAIL;
```

lpszKeys
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 no keys of this type are present on the device.

lpwszUNICODEKeys
String which holds the numeric and alphanumeric keys on the Text Terminal Unit like *lpszKeys* 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.

5. Execute Commands

5.1 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 or more of the following flags of type A, or 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 an exclamation signal.	B
WFS_TTU_BEEPWARNING	The beeper sounds a warning signal.	B
WFS_TTU_BEEPERROR	The beeper sounds an 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 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 None.

5.2 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. 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.

5.3 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 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 None.

5.4 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 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_INVALIDLED	An attempt to set a LED to a new value was invalid because the LED does not exist.

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

Comments None.

5.5 WFS_CMD_TTU_SET_RESOLUTION

Description This command is used to set the resolution of the display. The screen is cleared and the cursor is positioned at the upper left position.

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 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_RESNOTSUPP	The specified resolution is not supported by the display.

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

Comments None.

5.6 WFS_CMD_TTU_WRITE_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 LPWFSTTUWRITEFORM lpWriteform;

```
typedef struct _wfs_ttu_write_form
     {
     LPSTR               lpszFormName;
     BOOL                bClearScreen;
     LPSTR               lpszFields;
     LPWSTR              lpszUNICODEFields;
     } WFSTTUWRITEFORM, *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_ INF_TTU_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 *lpszFields* member is invalid.
WFS_ERR_TTU_CHARSETDATA	Character set(s) supported by Service Provider is inconsistent with use of *lpszFields* or *lpszUNICODEFields* 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.

5.7 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. The fields are edited by the user in the order that the fields are specified within this parameter. If *lpszFieldNames* value is a NULL pointer, then data is read from all input fields on the form in the order they appear in the form file (independent of the field screen position).

Output Param LPWFSTTUREADFORMOUT lpReadFormOut;

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

lpszFields
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 form is encoded in UNICODE.

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) read from the text terminal unit key pad for this field. This parameter is a NULL pointer if the form is encoded in 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 *lpszFieldNames* 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 The WFS_TTU_CK_ENTER key only acts as terminate key when it is pressed in the last read field. When the WFS_TTU_CK_ENTER key is pressed in an intermediate field, the cursor moves

to the next field and the data entry finishes for the current field. Any other key that terminates input (except cancel), will cause all the fields to be returned in their present state. If cancel terminates input then the command will return the WFS_ERR_TTU_KEYCANCELED error.

The following keys will not be returned in the output parameter *lpszFields* or *lpszUNICODEFields*, but they may affect the field content (note in the following the term *field content* is used to refer to the data buffer and the display field):

Value	Meaning
WFS_TTU_CK_CLEAR	Will clear the field content.
WFS_TTU_CK_BACKSPACE	Will cause the character before the Current Edit Position to be removed from the field content. If WFS_TTU_CK_BACKSPACE is the first key pressed after a field is activated (for any reason other than when the WFS_TTU_CK_BACKSPACE key causes the field to be activated), then the last character in the field content is deleted. If WFS_TTU_CK_BACKSPACE is pressed when the Current Edit Position is at the start of a field, then the previous field is activated. If WFS_TTU_CK_BACKSPACE is the first key pressed after the field is activated as a result of an earlier WFS_TTU_CK_BACKSPACE then no characters are deleted from the field content and the previous field will be activated. It is not possible to navigate backwards past the first field; in this case WFS_TTU_CK_BACKSPACE will have no effect.
WFS_TTU_CK_00	Will add a double zero ‘00’ string to the field content. If there is not enough space for all the digits to be added to the field content when the field’s OVERFLOW definition is TERMINATE or TRUNCATE then the excess ‘0’s will be ignored. If the field’s OVERFLOW definition is OVERWRITE then all the ‘0’s are added to the field content.
WFS_TTU_CK_000	Will add a triple zero ‘000’ string to the field content. If there is not enough space for all the digits to be added to the field content when the field’s OVERFLOW definition is TERMINATE or TRUNCATE then the excess ‘0’s will be ignored. If the field’s OVERFLOW definition is OVERWRITE then all the ‘0’s are added to the field content.

5.8 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;
     } 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
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 *lpsText* or *lpsUNICODEText* fields.

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

Comments None.

5.9 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. 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               lpszActiveKeys;
     LPWSTR              lpwszActiveUNICODEKeys;
     LPWORD              lpwActiveCommandKeys;
     LPWORD              lpwTerminateCommandKeys;
     } WFSTTUREAD, *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 *wPosX* and *wPosY*.

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 no keys of this type are active keys. *lpszActiveKeys* and *lpwszActiveUNICODEKeys* are mutually exclusive, so *lpszActiveKeys* must be a NULL pointer if *lpwszActiveUNICODEKeys* is not a NULL pointer.

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. *lpszActiveKeys* and *lpwszActiveUNICODEKeys* are mutually exclusive, so *lpwszActiveUNICODEKeys* must be a NULL pointer if *lpszActiveKeys* is not a NULL pointer.

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              lpszUNICODEInput;
} WFSTTUREADIN, *LPWFSTTUREADIN;
```

lpszInput
Specifies a zero terminated string containing all the printable characters (numeric and alphanumeric) read from the text terminal unit key pad.

lpszUNICODEInput
Specifies a zero terminated string containing all the printable characters (numeric and alphanumeric) read from the text terminal unit key pad.

Note 1: *lpszInput* and *lpszUNICODEInput* are mutually exclusive, so if *lpszInput* is not a NULL pointer then *lpszUNICODEInput* must be a NULL pointer, and vice versa.

Note 2: The following keys will not be returned in the output parameter *lpszInput* or *lpszUNICODEInput*, but they may affect the buffer if active:

Value	Meaning
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. If the WFS_TTU_CK_00 key is pressed, and there is not enough space for all the digits to be added to the buffer, then the key press will be ignored, no digits will be added to the buffer and no WFS_EXEE_TTU_KEY event will be generated.
WFS_TTU_CK_000	Will add a triple zero ‘000’ string to the buffer. If the WFS_TTU_CK_000 key is pressed, and there is not enough space for all the digits to be added to the buffer, then the key press will be ignored, no digits will be added to the buffer and no WFS_EXEE_TTU_KEY event will be generated.

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.

5.10 WFS_CMD_TTU_RESET

Description	Sends a service reset to the Service Provider. This command clears the screen, clears the keyboard buffer, sets the default resolution and sets the cursor position to the upper left.
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.

5.11 WFS_CMD_TTU_DEFINE_KEYS

Description This command defines the keys that will be active during the next WFS_CMD_TTU_READ_FORM command. The configured set will be active until the next WFS_CMD_TTU_READ_FORM command ends, at which point the default values are restored.

Input Param LPWFSTTUDEFKEYS lpDefKeys;

```
typedef struct _wfs_ttu_def_keys
     {
     LPSTR               lpszActiveKeys;
     LPWSTR              lpwszActiveUNICODEKeys;
     LPWORD              lpwActiveCommandKeys;
     LPWORD              lpwTerminateCommandKeys;
     } WFSTTUDEFKEYS, *LPWFSTTUDEFKEYS;
```

lpszActiveKeys
String which specifies the alphanumeric keys on the Text Terminal Unit, e.g. “12ABab”, to be active during the execution of the next WFS_CMD_TTU_READ_FORM 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 no keys of this type are active keys. *lpszActiveKeys* and *lpwszActiveUNICODEKeys* are mutually exclusive, so *lpszActiveKeys* must be a NULL pointer if *lpwszActiveUNICODEKeys* is not a NULL pointer.

lpwszActiveUNICODEKeys
String which specifies the alphanumeric keys on the Text Terminal Unit, e.g. “12ABab” (UNICODE), to be active during the execution of the next WFS_CMD_TTU_READ_FORM 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. *lpszActiveKeys* and *lpwszActiveUNICODEKeys* are mutually exclusive, so *lpwszUNICODEActiveKeys* must be a NULL pointer if *lpszActiveKeys* is not a NULL pointer.

lpwActiveCommandKeys
Array specifying the command keys which are active during the execution of the next WFS_CMD_TTU_READ_FORM 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 next WFS_CMD_TTU_READ_FORM 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 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_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 Only the generic error codes defined in [Ref. 1] can be generated by this command.

Comments None.

5.12 WFS_CMD_TTU_POWER_SAVE_CONTROL

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

Input Param LPWFSTTUPOWERSAVECONTROL lpPowerSaveControl;

```
typedef struct _wfs_ttu_power_save_control
     {
     USHORT               usMaxPowerSaveRecoveryTime;
     } WFSTTUPOWERSAVECONTROL, *LPWFSTTUPOWERSAVECONTROL;
```

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

Output Param None.

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

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

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

Value	Meaning
WFS_SRVE_TTU_POWER_SAVE_CHANGE	The power save recovery time has changed.

Comments None.

5.13 WFS_CMD_TTU_SET_LED_EX

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

For backwards compatibility the WFS_CMD_TTU_SET_LED command is provided.

Input Param LPWFSTTUSETLEDSEX lpSetLEDs;

```
typedef struct _wfs_ttu_set_leds_ex
     {
     USHORT               usLED;
     DWORD                dwCommand;
     } WFSTTUSETLEDSEX, *LPWFSTTUSETLEDSEX;
```

usLED
Specifies the index (zero to *usNumOfLEDs*-1 as reported in WFS_INF_TTU_CAPABILITIES) of the LED to set as one of the values defined within the capabilities section.

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

Value	Meaning	Type
WFS_TTU_LEDOFF	The LED is turned off.	A
WFS_TTU_LEDSLOWFLASH	The LED is set to flash slowly.	B
WFS_TTU_LEDMEDIUMFLASH	The LED is set to flash medium frequency.	B
WFS_TTU_LEDQUICKFLASH	The LED is set to flash quickly.	B
WFS_TTU_LEDCONTINUOUS	The LED is turned on continuously (steady).	B
WFS_TTU_LEDRED	The LED color is set to red.	C
WFS_TTU_LEDGREEN	The LED color is set to green.	C
WFS_TTU_LEDYELLOW	The LED color is set to yellow.	C
WFS_TTU_LEDBLUE	The LED color is set to blue.	C
WFS_TTU_LEDCYAN	The LED color is set to cyan.	C
WFS_TTU_LEDMAGENTA	The LED color is set to magenta.	C
WFS_TTU_LEDWHITE	The LED color is set to white.	C

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_INVALIDLED	An attempt to set an LED to a new value was invalid because the LED does not exist.

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

Comments None.

6. Events

6.1 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                lpszFormName;
     LPSTR                lpszFieldName;
     WORD                 wFailure;
     } WFSTTUFIELDFAIL, *LPWFSTTUFIELDFAIL;
```

lpszFormName
Points to the null-terminated form name.

lpszFieldName
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 must be supplied by the application.
WFS_TTU_FIELDSTATICOVWR	The specified field is static and thus cannot 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.

6.2 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.

6.3 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;
     } WFSTTUKEY, *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.

6.4 WFS_SRVE_TTU_DEVICEPOSITION

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

Event Param LPWFSTTUDEVICEPOSITION lpDevicePosition;

```
typedef struct _wfs_ttu_device_position
     {
     WORD                wPosition;
     } WFSTTUDEVICEPOSITION, *LPWFSTTUDEVICEPOSITION;
```

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

Value	Meaning
WFS_TTU_DEVICEINPOSITION	The device is in its normal operating position.
WFS_TTU_DEVICENOTINPOSITION	The device has been removed from its normal operating position.
WFS_TTU_DEVICEPOSUNKNOWN	The position of the device cannot be determined.

Comments None.

6.5 WFS_SRVE_TTU_POWER_SAVE_CHANGE

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

Event Param LPWFSTTUPOWERSAVECHANGE lpPowerSaveChange;

```
typedef struct _wfs_ttu_power_save_change
     {
     USHORT              usPowerSaveRecoveryTime;
     } WFSTTUPOWERSAVECHANGE, *LPWFSTTUPOWERSAVECHANGE;
```

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

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

7. 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.

7.1 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 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.
- In the form definition file, where characters are expressed using standard C hexadecimal escape sequences, the high order byte is defined first. For example, "\x0041" would represent the character 'A'. This is independent of the encoding format of the form definition file.

7.2 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.

7.3 Form Definition [1]

XFSFORM		*formname**	
BEGIN			
(required)	**SIZE**	*width*, *height*	Width of form Height of form
	VERSION	*major*, *minor*, *date**, *author**	Major version number (default 0) Minor version number (default 0) 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			

[1] Attributes are not required in any mandatory order within a Form Definition.

7.4 Field Definition [2]

XFSFIELD		*fieldname**	
BEGIN			
	LANGUAGE	*languageID*	Language used for this field. See Form definition for detailed description. If unspecified defaults to form definition LANGUAGE specification.
(required)	**POSITION**	*x,* *y*	Horizontal position (relative to left side of form) Vertical position (relative to top of form) The initial left upper position is referenced as (0,0)
(required)	**SIZE**	*width,* *height*	Field width Field height
	TYPE	*fieldtype*	Type of field: TEXT (default) INVISIBLE PASSWORD (contents is echoed with '*') GRAPHIC (ignored for WFS_CMD_TTU_READ_FORM commands)
	SCALING	*scalingtype*	Information on how to size the GRAPHIC within the field: BESTFIT (default) scale to size indicated ASIS render at native size MAINTAINASPECT scale as close as possible to size indicated while maintaining the aspect ratio and not losing graphic information. SCALING is only relevant for GRAPHICS field types
	CLASS	*class*	Field class: OPTIONAL (default) STATIC REQUIRED
	KEYS	*keys*	Accepted input key types: NUMERIC HEXADECIMAL ALPHANUMERIC This is an optional field where the default value is vendor dependent.
	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

[2] Attributes are not required in any mandatory order within a Field Definition.

	FORMAT	*formatstring* *	This is an application defined input field describing how the application should format the data. This may be interpreted by the Service Provider.
	INITIALVALUE	*value**	Initial value. For GRAPHIC type fields, this value will contain the filename of the graphic image. The type of this graphic will be determined by the file extension (e.g. BMP for Windows Bitmap). The graphic file name must contain the full path. For example “C:\XFS\BSVCLOGO.BMP” illustrates the use of the full path name
END			

8. C - Header file

```
/******************************************************************************
*                                                                             *
* xfsttu.h      XFS - Text Terminal Unit (TTU) definitions                    *
*                                                                             *
*               Version 3.20  (March 02 2011)                                 *
*                                                                             *
******************************************************************************/

#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       (0x1403) /* Version 3.20 */

#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_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)
#define     WFS_CMD_TTU_DEFINE_KEYS             (TTU_SERVICE_OFFSET + 11)
#define     WFS_CMD_TTU_POWER_SAVE_CONTROL      (TTU_SERVICE_OFFSET + 12)
#define     WFS_CMD_TTU_SET_LED_EX              (TTU_SERVICE_OFFSET + 13)

/* 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)
#define    WFS_SRVE_TTU_DEVICEPOSITION          (TTU_SERVICE_OFFSET + 4)
#define    WFS_SRVE_TTU_POWER_SAVE_CHANGE       (TTU_SERVICE_OFFSET + 5)

/* 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
```

```
#define     WFS_TTU_DEVFRAUDATTEMPT             WFS_STAT_DEVFRAUDATTEMPT
#define     WFS_TTU_DEVPOTENTIALFRAUD           WFS_STAT_DEVPOTENTIALFRAUD

/* 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
             WFSTTUSTATUS.lpLEDEx.lpdwLEDs
             WFSTTUCAPS.lpLEDEx.lpdwLEDs
             WFSTTUSETLEDS.fwCommand */

#define     WFS_TTU_LEDNA                       (0x0000)
#define     WFS_TTU_LEDOFF                      (0x0001)
#define     WFS_TTU_LEDSLOWFLASH                (0x0002)
#define     WFS_TTU_LEDMEDIUMFLASH              (0x0004)
#define     WFS_TTU_LEDQUICKFLASH               (0x0008)
#define     WFS_TTU_LEDCONTINUOUS               (0x0080)

/* Values of WFSTTUSTATUS.lpLEDEx.lpdwLEDs
             WFSTTUCAPS.lpLEDEx.lpdwLEDs
             WFSTTUSETLEDSEX.dwCommand */

#define     WFS_TTU_LEDRED                      (0x00000100)
#define     WFS_TTU_LEDGREEN                    (0x00000200)
#define     WFS_TTU_LEDYELLOW                   (0x00000400)
#define     WFS_TTU_LEDBLUE                     (0x00000800)
#define     WFS_TTU_LEDCYAN                     (0x00001000)
#define     WFS_TTU_LEDMAGENTA                  (0x00002000)
#define     WFS_TTU_LEDWHITE                    (0x00004000)

/* Values of WFSTTUSTATUS.wDevicePosition
             WFSTTUDEVICEPOSITION.wPosition */

#define     WFS_TTU_DEVICEINPOSITION            (0)
#define     WFS_TTU_DEVICENOTINPOSITION         (1)
#define     WFS_TTU_DEVICEPOSUNKNOWN            (2)
#define     WFS_TTU_DEVICEPOSNOTSUPP            (3)

/* values of WFSTTUSTATUS.wAntiFraudModule */

#define     WFS_TTU_AFMNOTSUPP                  (0)
#define     WFS_TTU_AFMOK                       (1)
#define     WFS_TTU_AFMINOP                     (2)
#define     WFS_TTU_AFMDEVICEDETECTED           (3)
#define     WFS_TTU_AFMUNKNOWN                  (4)

/* 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 WFSTTUKEYDETAIL.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))
#define WFS_ERR_TTU_POWERSAVETOOSHORT           (-(TTU_SERVICE_OFFSET + 15))

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

typedef struct _wfs_ttu_led_ex
{
    USHORT                  usNumOfLEDs;
    LPDWORD                 lpdwLEDs;
} WFSTTULEDEX, *LPWFSTTULEDEX;

typedef struct _wfs_ttu_status
{
    WORD                    fwDevice;
    WORD                    wKeyboard;
    WORD                    wKeylock;
    WORD                    wLEDs[WFS_TTU_LEDS_MAX];
    WORD                    wDisplaySizeX;
    WORD                    wDisplaySizeY;
```

```
    LPSTR                lpszExtra;
    WORD                 wDevicePosition;
    USHORT               usPowerSaveRecoveryTime;
    LPWFSTTULEDEX        lpLEDEx;
    WORD                 wAntiFraudModule;
} 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;
    BOOL                 bPowerSaveControl;
    LPWFSTTULEDEX        lpLEDEx;
    BOOL                 bAntiFraudModule;
} WFSTTUCAPS, *LPWFSTTUCAPS;

typedef struct _wfs_ttu_frm_header
{
    LPSTR                lpszFormName;
    WORD                 wWidth;
    WORD                 wHeight;
    WORD                 wVersionMajor;
    WORD                 wVersionMinor;
    WORD                 fwCharSupport;
    LPSTR                lpszFields;
    WORD                 wLanguageID;
} 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;
    WORD                 wLanguageID;
} WFSTTUFRMFIELD, *LPWFSTTUFRMFIELD;

typedef struct _wfs_ttu_key_detail
{
    LPSTR                lpszKeys;
    LPWSTR               lpwszUNICODEKeys;
    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                  lpszUNICODEFields;
} WFSTTUWRITEFORM, *LPWFSTTUWRITEFORM;

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

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

typedef struct _wfs_ttu_def_keys
{
    LPSTR                   lpszActiveKeys;
    LPWSTR                  lpwszActiveUNICODEKeys;
    LPWORD                  lpwActiveCommandKeys;
    LPWORD                  lpwTerminateCommandKeys;
} WFSTTUDEFKEYS, *LPWFSTTUDEFKEYS;

typedef struct _wfs_ttu_write
{
    WORD                    fwMode;
    SHORT                   wPosX;
    SHORT                   wPosY;
    WORD                    fwTextAttr;
    LPSTR                   lpsText;
    LPWSTR                  lpsUNICODEText;
} 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                  lpwszActiveUNICODEKeys;
    LPWORD                  lpwActiveCommandKeys;
    LPWORD                  lpwTerminateCommandKeys;
} WFSTTUREAD, *LPWFSTTUREAD;

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

typedef struct _wfs_ttu_power_save_control
{
```

```
    USHORT                usMaxPowerSaveRecoveryTime;
} WFSTTUPOWERSAVECONTROL, *LPWFSTTUPOWERSAVECONTROL;

typedef struct _wfs_ttu_set_leds_ex
{
    USHORT                usLED;
    DWORD                 dwCommand;
} WFSTTUSETLEDSEX, *LPWFSTTUSETLEDSEX;

/*=================================================================*/
/* 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;

typedef struct _wfs_ttu_device_position
{
    WORD                  wPosition;
} WFSTTUDEVICEPOSITION, *LPWFSTTUDEVICEPOSITION;

typedef struct _wfs_ttu_power_save_change
{
    USHORT                usPowerSaveRecoveryTime;
} WFSTTUPOWERSAVECHANGE, *LPWFSTTUPOWERSAVECHANGE;

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

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

#endif /* __INC_XFSTTU__H */
```