CEN

WORKSHOP

AGREEMENT

CWA 13449-7

December 1998

ICS 35.200;35.240.40

English version

Extensions for Financial Services (XFS) interface specification - Part 7: Check Reader/Scanner Device Class Interface - Programmer's Interface

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

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

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

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

CEN Members are the National Standards Bodies of Austria, Belgium, Czech Republic, Denmark, Finland, France, Germany, Greece, Iceland, Ireland, Italy, Luxembourg, Netherlands, Norway, Portugal, Spain, Sweden, Switzerland and United Kingdom.

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

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

Ref. No. CWA 13449-7:1998 E

Contents

Foreword

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

This specification was originally developed by the Banking Solutions Vendor Council (BSVC), and is endorsed by the CEN/ISSS Workshop on XFS. This Workshop gathers both suppliers (among others the BSVC members) as well as banks and other financial service companies. A list of companies participating in this Workshop and in support of this CWA is available from the CEN/ISSS Secretariat.

The specification is continuously reviewed and commented in the CEN/ISSS Workshop on XFS. It is therefore expected that an update of the specification will be published in due time as a CWA, superseding this revision 2.00.

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

0. Introduction

This is part 7 of the multi-part CWA 13449, describing Release 2.0 of the XFS interface specification.

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

Part 1: Application Programming Interface (API) - Service Provider Interface (SPI); Programmer's Reference
Part 2: Service Classes Definition; Programmer's Reference
Part 3: Printer Device Class Interface - Programmer's Reference
Part 4: Identification Card Device Class Interface - Programmer's Reference
Part 5: Cash Dispenser Device Class Interface - Programmer's Reference
Part 6: PIN Keypad Device Class Interface - Programmer's Reference
Part 7: Check Reader/Scanner Device Class Interface - Programmer's Reference
Part 8: Depository Device Class Interface - Programmer's Reference
Part 9: Text Terminal Unit Device Class Interface - Programmer's Reference
Part 10: Sensors and Indicators Unit Device Class Interface - Programmer's Reference
Part 11: Vendor Dependent Mode Device Class Interface - Programmer's Reference
Part 12: Camera Device Class Interface - Programmer's Reference

In addition to these Programmer's Reference specifications, the reader of this CWA is also referred to a complementary document, called Release Notes. The Release Notes contain clarifications and explanations on the CWA specifications, which are not requiring functional changes. The current version of the Release Notes is available from the CEN/ISSS Secretariat (contact isss@cenorm.be or download from http://www.cenorm.be/isss/ Workshop/XFS/release-notes.htm).

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

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

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

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

Revision History:

1.0	**May 24, 1993**	**Initial release of API and SPI specification**
1.11	**February 3, 1995**	**Separation of specification into separate documents for API/SPI and service class definitions, with updates**
2.00	**November 11, 1996**	**Updated release encompassing self-service environment.**
	October 6, 1998	**WOSA/XFS Release 2.00 as originally developed by the BSVC, has been formally accepted as a CEN Workshop Agreement by the CEN/ISSS XFS Workshop and the name WOSA/XFS has been changed into XFS. In spite of the name change, certain occurrencies of WOSA/XFS however still appear in the documentation, for compatibility reasons**

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

1. XFS Service-Specific Programming

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

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

In general, the specific command set for a service class is defined as the union of the sets of 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 command set defined for the class.

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. Check Readers and Scanners

This specification describes the XFS service class of check readers and scanners. Check image scanners are treated as a special case of check readers, i.e., image-enabled instances of the latter. This class includes devices with a range of features, from small hand-held read-only devices through which checks are manually swiped one at a time, to much larger devices (i.e., tabletop) which automatically feed checks by the batch past a reader, an encoder, an endorser, an optional image scanner, to be sorted into one of several pockets. The high end device of this class usually found in bank branches shares many capabilities with the still larger devices usually found only in a bank's central data processing site (i.e., high-speed reader/sorters), but the latter are not explicitly addressed here. The specification of this service class includes definitions of the service-specific commands that can be issued, using the **WFSAsyncExecute**, **WFSExecute, WFSGetInfo** and **WFSAsyncGetInfo** functions.

In the U.S., checks are always encoded in magnetic ink for reading by Magnetic Ink Character Recognition (MICR), and a single font is always used. In Europe some countries use MICR and some use Optical Character Recognition (OCR) character sets, with different fonts, for their checks.

In all countries, typical fields found encoded on a check include the bank ID number and the account number. Part of the processing done by the bank is to also encode the amount on the check, usually done by having an operator enter the handwritten or typewritten face amount on a numeric keypad.

3. Info Commands

3.1 WFS_INF_CHK_STATUS

Description This function is used to query the status of the device and the service.

Input Param None.

Output Param

```
LPWFSCHKSTATUS   lpStatus;

struct _wfs_chk_status
   {
   WORD          fwDevice;
   WORD          fwMedia;
   WORD          fwInk;
   DWORD         dwMode;
   WORD          fwLamp;
   LPSTR         lpszExtra;
   } WFSCHKSTATUS, * LPWFSCHKSTATUS;
```

fwDevice
Specifies the state of the check reader device as one of:

Value	Meaning
WFS_CHK_DEVONLINE	Device is online.
WFS_CHK_DEVOFFLINE	Device is offline.
WFS_CHK_DEVPOWEROFF	Device is powered off.
WFS_CHK_DEVNODEVICE	No device is connected.

fwMedia
Specifies the status of the media in the check reader as one of:

Value	Meaning
WFS_CHK_MEDIANOTPRESENT	No media is inserted in device.
WFS_CHK_MEDIAREQUIRED	Insertion of media required.
WFS_CHK_MEDIAPRESENT	Media inserted in device.
WFS_CHK_MEDIAJAMMED	Media jam in device.

fwInk
Specifies the status of the ink in the check reader as one of:

Value	Meaning
WFS_CHK_INKFULL	Ink supply in device is full.
WFS_CHK_INKLOW	Ink supply in device is low.
WFS_CHK_INKOUT	Ink supply in device is empty.

dwMode
Specifies the autofeed status of the check reader as one of:

Value	Meaning
WFS_CHK_MODEMANUAL	Device is in manual mode.
WFS_CHK_MODEAUTOFEED	Device is in autofeed mode.

fwLamp
Specifies the status of the check reader imaging lamp as one of:

Value	Meaning
WFS_CHK_LAMPOK	The lamp is OK.
WFS_CHK_LAMPFADING	The lamp should be changed.

lpszExtra

Points to a list of vendor-specific, or any other extended information. The information is returned as a series of *"key=value"* strings so that it is easily extensible by service providers. Each string is null-terminated, with the final string terminating with two null characters.

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

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

3.2 WFS_INF_CHK_CAPABILITIES

Description This function is used to request device capability information.

Input Param None.

Output Param

```
LPWFSCHKCAPS  lpCaps;

typedef struct _wfs_chk_caps
   {
   WORD      wClass;
   WORD      fwType;
   BOOL      bCompound;
   BOOL      bMICR;
   BOOL      bOCR;
   BOOL      bAutoFeed;
   BOOL      bEndorser;
   BOOL      bEncoder;
   WORD      fwStamp;
   WORD      wImageCapture;
   USHORT    usPockets;
   LPSTR     lpszFontNames;
   LPSTR     lpszEncodeNames;
   LPSTR     lpszExtra;
   }  WFSCHKCAPS, * LPWFSCHKCAPS;
```

fwClass
Specifies the logical service; value is WFS_SERVICE_CLASS_CHK.

fwType
Specifies the type of the physical device; only current value is WFS_CHK_TYPECHK.

bCompound
TRUE if the logical device is part of a compound device.

bMICR
TRUE if the device can read MICR on checks.

bOCR
*TRUE if the device c*an read OCR on checks.

bAutoFeed
*TRUE if the device h*as autofeed capability; FALSE if only manual feed.

bEndorser
TRUE if a programmable endorser is present.

bEncoder
*TRUE if a*n encoder is present.

fwStamp
One of:

Value	Meaning
WFS_CHK_STAMPNONE	Device cannot stamp/endorse check
WFS_CHK_STAMPFRONT	Device can stamp/endorse front of check
WFS_CHK_STAMPREAR	Device can stamp/endorse back of check
WFS_CHK_STAMPBOTH	Device can stamp/endorse both sides

wImageCapture
Uses same values as *wStamp* to indicate from which sides of a check the device can capture images.

usPockets
Number of pockets; if 0 or 1, device has no pockets.

lpszFontNames
The names of the fonts supported for reading; each is terminated with a NULL and the string is terminated with two NULLs.

lpszEncodeNames
The names of the fonts supported for encoding; each is terminated with a NULL and the string is terminated with two NULLs.

lpszExtra
Points to a list of vendor-specific, or any other extended information. The information is returned as a series of "*key=value*" strings so that it is easily extensible by service providers. Each string is null-terminated, with the final string terminating with two null characters.

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

Comments The font names are standardized so that applications can check for standard literals, e.g.: CMC7, E13B. Reserved OCR font names are TBD due to numerous local variants. (i.e. OCRA and OCRB are not enough).

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

3.3 WFS_INF_CHK_FORM_LIST

Description This function is used to retrieve the list of forms available to the service.

Input Param None.

Output Param
```
LPSTR       lpszFormList;
```

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

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

3.4 WFS_INF_CHK_QUERY_FORM

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

Input Param
```
LPSTR       lpszFormName;
```

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

Output Param
```
LPWFSFRMHEADER
```

See section 7.1.4.5 WFS_INF_PTR_QUERY_FORM, for details of this structure.

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

Value	Meaning
WFS_ERR_CHK_FORMNOTFOUND	The specified form cannot be found.

3.5 WFS_INF_CHK_QUERY_FIELD

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

Input Param `LPWFSCHKQUERYFIELD`, as defined below.

```
typedef struct _wfs_chk_query_field
   {
   LPSTR            lpszFormName;
   LPSTR            lpszFieldName;
   } WFSCHKQUERYFIELD, * LPWFSCHKQUERYFIELD;
```

lpszFormName
Points to the null-terminated form name.

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

Output Param `LPWFSFRMFIELD * lpFields;`

See Section 7.1.4.7, WFS_PTR_QUERY_FIELD for details of this structure.

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

Value	Meaning
WFS_ERR_CHK_FORMNOTFOUND	The specified form cannot be found.
WFS_ERR_CHK_FIELDNOTFOUND	The specified field cannot be found.

4. Execute Commands

4.1 WFS_CMD_CHK_READ_FORM

Description This function returns the data from the current check. The contents of all the fields within the form are returned to the application. For small hand-held check readers, this command might be the only one used.

Input Param LPWFSCHKINREADFORM

```
typedef struct _wfs_chk_in_read_form
   {
   LPSTR     lpszFormName;
   LPSTR     lpszFieldNames;
   DWORD     dwOptions;
   LPSTR     lpszExtra;
   }  WFSCHKINREADFORM, * LPWFSCHKINREADFORM;
```

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

lpszFieldNames
Points to a list of NULL-terminated field names from which to read input data, with the final name terminating with two NULLs.

dwOptions
WFS_CHK_OPTAUTOFEED

lpszExtra
Points to a list of vendor-specific, or any other extended information. The information is returned as a series of "*key=value*" strings so that it is easily extensible by service providers. Each string is null-terminated, with the final string terminating with two null characters.

Output Param LPWFSCHKOUTREADFORM

```
typedef struct _wfs_chk_out_read_form
   {
   WORD      hDoc;
   LPSTR     lpszFields;
   }  WFSCHKOUTREADFORM, * LPWFSCHKOUTREADFORM;
```

hDoc
Handle to this check.

lpszFields
Points to a list of field data returned. See Comments.

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

Value	Meaning
WFS_ERR_CHK_REQDFIELDMISSING	The check was blank.
WFS_ERR_CHK_FORMNOTFOUND	Invalid form name.
WFS_ERR_CHK_FIELDSPECFAILURE	The syntax of the *lpszFields* member is invalid.
WFS_ERR_CHK_INCOMPLETEREAD	Read errors occurred and an incomplete code line is available. Question marks are returned in place of any numbers which could not be read. A code line will always be returned when this error occurs, and the application may choose different behavior depending on the number of question marks returned, e.g., prompt the operator to enter missing numbers.

Execute Events The following execute events can be generated by this command:

Value	Meaning
WFS_EXEE_CHK_NOMEDIA	No check has been inserted in the (manual mode) check reader; to be used by the application to generate a message to the operator to insert a check.
WFS_EXEE_CHK_MEDIAINSERTED	A check was inserted; this is only issued following the above event.

Comments. At the end of a successful WFS_CMD_CHK_READ_FORM, the string pointed to by *lpsFields* will contain a sequence such as (given a U.S. personal check):

```
ROUTETRANS=021203501\0ACCOUNT=370361\0TRANCODE=2199\0AMOUNT=0000001000\0\0
```

Each *fieldname=value* pair is terminated by a NULL; the end of the buffer is marked with an additional NULL. Any embedded space characters (0x20) are significant; trailing spaces are not.

The timeout parameter (*dwTimeOut*) in the **WFSExecute** request that passes this command should always be large enough to accomodate prompting the operator to insert a check, having the operator do so, and processing the check. If the timeout expires before these operations are completed, the **WFSExecute** will be canceled, possibly leaving an application-generated prompt on the operator's screen.

4.2 WFS_CMD_CHK_MULTICOMMAND

Description This function is used to encode the amount field of the check, optionally stamp and endorse the check, and select a pocket to which the check will be sorted if the device supports these capabilities.

Input Param `LPWFSCHKMULTICOMMAND`

```
typedef struct _wfs_chk_multicommand
   {
   WORD      hDoc;
   DWORD     dwOptions;
   BYTE      bPocket;
   LPSTR     lpszEncodeFormName;
   LPSTR     lpszEncodeFields;
   LPSTR     lpszEndorserFormName;
   LPSTR     lpszEndorserFields;
   LPSTR     lpszExtra;
   } WFSCHKMULTICOMMAND, * LPWFSCHKMULTICOMMAND;
```

hDoc
handle to the check to be processed; NULL means "current" check.

dwOptions
Command options, as a combination of the following flags:
WFS_CHK_OPTSTAMPFRONT
WFS_CHK_OPTSTAMPBACK
WFS_CHK_OPTENDORSEFRONT
WFS_CHK_OPTENDORSEBACK
WFS_CHK_OPTSORTONLY
WFS_CHK_OPTTAKEIMAGE

bPocket
Ignored if no sorter present.

lpszEncodeFormName
Name of form defining encoder fields.

lpszEncodeFields
List of fieldname/value pairs for encoder.

lpszEndorserFormName
Name of form defining endorser fields.

lpszEndorserFields
List of fieldname/value pairs for endorser.

lpszExtra
Points to a list of vendor-specific, or any other extended information. The information is returned as a series of "*key=value*" strings so that it is easily extensible by service providers. Each string is null-terminated, with the final string terminating with two null characters.

Output Param None.

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

Value	Meaning
WFS_ERR_CHK_FORMNOTFOUND	Invalid form name.
WFS_ERR_CHK_FIELDNOTFOUND	Invalid field name.
WFS_ERR_CHK_REQDFIELDMISSING	A field required by the form is not supplied.
WFS_ERR_CHK_EXTRAFIELD	A field supplied by the application does not exist in this form (warning).
WFS_ERR_CHK_FIXEDOVERWRITE	The application passed a field which is marked as fixed in the form description (warning).
WFS_ERR_CHK_FIELDSPECFAILURE	The syntax of the *lpszFields* member is invalid.
WFS_ERR_CHK_UNSUPPORTEDCAP	The service does not have a capability requested in this command (i.e. a pocket sort was requested on a device with zero pockets). This is a warning; the requested capability is ignored.

Execute Events

WFS_EXEE_CHK_NOMEDIA	No check has been inserted in the (manual mode) check reader.
WFS_EXEE_CHK_MEDIAINSERTED	A check was inserted; this is only issued following the above event.

Comments The contents of the *lpszFields* parameter is as follows:

```
fieldname=value\0fieldname=value\0.......fieldname=value\0\0
```

Each *fieldname=value* pair is terminated with a NULL; the end of the buffer is marked with an additional NULL.

If an extra field is passed to the command verb a warning message will be returned. If a required field is missing an error message is returned and the form is not printed. Missing optional fields don't cause a problem. Overwriting of a fixed field results in an error and the print operation does not occur.

The *lpszEncodeFormName* parameter should be the same as the form name used previously to read the encode line with WFS_CMD_CHK_READ_FORM. Results are unpredictable if a different form name is used.

4.3 WFS_CMD_CHK_READ_IMAGE

Description This function returns image data from the current check in TIFF 6.0 format.

Input Param `LPWFSCHKINREADIMAGE`

```
typedef struct _wfs_chk_in_read_image
   {
   WORD      hDoc;
   DWORD     dwOptions;
   LPSTR     lpszExtra;
   } WFSCHKINREADIMAGE, * LPWFSCHKINREADIMAGE;
```

hDoc
Handle to the check whose image is to be returned.

DwOptions
[No options have been defined as of this revision.]

lpszExtra
Points to a list of vendor-specific, or any other extended information. The information is returned as a series of "*key=value*" strings so that it is easily extensible by service providers. Each string is null-terminated, with the final string terminating with two null characters.

Output Param `LPWFSCHKOUTREADIMAGE`

```
struct wfs_chk_out_read_image
   {
   WORD      wImage;
   LPSTR     lpImage;
   } WFSCHKOUTREADIMAGE, * LPWFSCHKOUTREADIMAGE;
```

wImage
Count of bytes of image data.

lpImage
Points to the image data.

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

Value	Meaning
WFS_ERR_CHK_INVALIDHDOC	*hDoc* is required but the value input does not correspond to a previously read document.
WFS_ERR_CHK_IMAGENOTAVAIL	The check referred to by *hDoc* does not have an image available.

Execute Events None.

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

4.4 WFS_CMD_CHK_MODE_SWITCH

Description This function is used to turn the autofeed mechanism off if it is running, or to turn it on if it is not.

Input Param

```
DWORD           dwMode;
```

dwMode
Autofeed mode specified as one of the following :

Value	Meaning
WFS_CHK_MODEMANUAL	Set device to manual if in autofeed mode
WFS_CHK_MODEAUTOFEED	Set device to autofeed if in manual mode

Output Param None.

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

Value	Meaning
WFS_ERR_CHK_INVALIDCOMMAND	The device does not support a mode switch.

Execute Events None.

Comments None.

5. Pragmatics of using the commands

This section discusses how the **WFSExecute** commands above map to the variety of check readers used in branch banking.

Small hand-held devices which contain only a MICR or an OCR reader, and through which checks are manually swiped, will normally be managed using only the WFS_CMD_CHK_READ_FORM command. Applications written for such devices can make sure that the check readers to which they are configured to attach are suitable by using the WFS_INF_CHK_CAPABILITIES command in **WFSGetInfo** to make sure that *fAutoFeed* is FALSE, *nPockets* is zero, and so on.

Applications written for table-top check readers with autofeed and/or sorting capability should ensure that the services to which they connect have the appropriate capabilities. The error WFS_ERR_UNSUPP_CATEGORY will be returned if the service does not have these capabilities. In many cases, the applications for such devices will have to run on the workstation to which the check reader is directly attached in order that the commands be able to keep up with the track through which the checks are moving.

6. Execute Events, Results, Codes

6.1 WFS_EXEE_CHK_NOMEDIA

Description This event specifies that the physical check must be inserted into the device in order for the execute command to proceed.

Event Param
```
LPSTR     lpszUserPrompt;
```

lpszUserPrompt
Points to a null-terminated string which identifies the prompt string which is configured for the form (the USERPROMPT attribute of the XFSFORM section).

Comments The application may use the *lpszUserPrompt* in any manner it sees fit. For example, it might display the string to the operator, along with a message that the check should be inserted.

6.2 WFS_EXEE_CHK_MEDIAINSERTED

Description This event specifies that the physical check has been inserted into the device.

Event Param None.

Comments The application may use this event to, for example, remove a message box from the screen telling the user to insert the next check.

7. Forms Language Usage

This section covers the usage of the forms language to accomodate check readers. The XFS forms language is defined in section 7.1.

The forms language contains the FORMAT attribute in the XFSFIELD section. For check readers, the *formatstring* is used to generate the delimiters for the check fields; its usage is ***not*** application-defined. The usage is the same for the check readers service class. For forms intended for use with check readers, the FORMAT attribute is required:

```
field  Amount                    FORMAT ":NNNNNNNNNN:"
field  AccountNum                FORMAT "0000NNNNNN<"
field  RouteTransit              FORMAT ";NNNNNNNNN;"
```

using punctuation in place of the standard field separators. A capital N means a number to be read and returned. A zero (“0”) means an optional number which, if present, is read and returned. Note that all fields on a check encoder line that have optional numbers should place the zeros on the same end of the format string as an aid to the Service Provider in parsing the code line (for instance, most check readers read the MICR line right to left, so optional numbers should always be on the left side of fields which have them.).

Normally, the format string, which gives the starting delimiter for each field, and the FOLLOWS clause, allow the service to parse the fields from the check's code line. The position attributes are used to specify the minimum and maximum starting locations for each field, so that a misread delimiter character can be detected and the parsing corrected (if the service is sophisticated enough to do this).

If the device supports reading multiple fonts, the FONT attribute of the XFSFIELD section might be significant. The name of the font (e.g. CMC7, E13B, etc), given here, will cause the check reader to use the appropriate font.

For endorsing checks, the field description specifies the “front” or “back” of the check using the SIDE attribute, and position relative to the trailing or (usually) leading edge of the check.

8. C-Header file

```
/******************************************************************************
*                                                                             *
* xfschk.h      XFS - Check reader/scanner (CHK) definitions                  *
*                                                                             *
*             Version 2.00  --  (01/20/97)                                    *
*                                                                             *
******************************************************************************/

#ifndef __INC_XFSCHK__H
#define __INC_XFSCHK__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

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


/* value of _wfs_chk_caps.wClass */

#define             WFS_SERVICE_CLASS_CHK                   (5)

#define             CHK_SERVICE_OFFSET                      (WFS_SERVICE_CLASS_CHK * 100)

/* CHK Info Commands */

#define             WFS_INF_CHK_STATUS                      (CHK_SERVICE_OFFSET + 1)
#define             WFS_INF_CHK_CAPABILITIES                (CHK_SERVICE_OFFSET + 2)
#define             WFS_INF_CHK_FORM_LIST                   (CHK_SERVICE_OFFSET + 3)
#define             WFS_INF_CHK_QUERY_FORM                  (CHK_SERVICE_OFFSET + 4)
#define             WFS_INF_CHK_QUERY_FIELD                 (CHK_SERVICE_OFFSET + 5)

/* CHK Command Verbs */

#define             WFS_CMD_CHK_READ_FORM                   (CHK_SERVICE_OFFSET + 1)
#define             WFS_CMD_CHK_MULTICOMMAND                (CHK_SERVICE_OFFSET + 2)
#define             WFS_CMD_CHK_READ_IMAGE                  (CHK_SERVICE_OFFSET + 3)
#define             WFS_CMD_CHK_MODE_SWITCH                 (CHK_SERVICE_OFFSET + 4)

/* CHK Messages */

#define             WFS_EXEE_CHK_NOMEDIA                    (CHK_SERVICE_OFFSET + 1)
#define             WFS_EXEE_CHK_MEDIAINSERTED              (CHK_SERVICE_OFFSET + 2)

/* values of _wfs_chk_status.fwDevice  */

#define             WFS_CHK_DEVONLINE                       (0)
#define             WFS_CHK_DEVOFFLINE                      (1)
#define             WFS_CHK_DEVPOWEROFF                     (2)
```

```
#define     WFS_CHK_DEVNODEVICE                 (3)

/* values of _wfs_chk_status.fwMedia */

#define     WFS_CHK_MEDIAPRESENT                (0)
#define     WFS_CHK_MEDIANOTPRESENT             (1)
#define     WFS_CHK_MEDIAREQUIRED               (2)
#define     WFS_CHK_MEDIAJAMMED                 (3)

/* values of _wfs_chk_status.fwInk */

#define     WFS_CHK_INKFULL                     (0)
#define     WFS_CHK_INKLOW                      (1)
#define     WFS_CHK_INKOUT                      (2)

/* values of _wfs_chk_status.dwMode, _wfs_in_mode_switch.dwMode */

#define     WFS_CHK_MODEMANUAL                  (0)
#define     WFS_CHK_MODEAUTOFEED                (1)

/* values of _wfs_chk_status.fwLamp */

#define     WFS_CHK_LAMPOK                      (0)
#define     WFS_CHK_LAMPFADING                  (1)

/* values of _wfs_chk_caps.fwStamp, _wfs_chk_caps.wImageCapture */

#define     WFS_CHK_STAMPNONE                       (1)
#define     WFS_CHK_STAMPFRONT                      (2)
#define     WFS_CHK_STAMPREAR                       (3)
#define     WFS_CHK_STAMPBOTH                       (4)

/* values of _wfs_in_multicommand.dwOptions */

#define     WFS_CHK_OPTSTAMPFRONT                   (1)
#define     WFS_CHK_OPTSTAMPBACK                    (2)
#define     WFS_CHK_OPTENDORSEFRONT                 (3)
#define     WFS_CHK_OPTENDORSEBACK                  (4)
#define     WFS_CHK_OPTSORTONLY                     (5)
#define     WFS_CHK_OPTTAKEIMAGE                    (6)

/* XFS CHK Errors */

#define  WFS_ERR_CHK_REQDFIELDMISSING           (-(CHK_SERVICE_OFFSET + 0))
#define  WFS_ERR_CHK_FORMNOTFOUND               (-(CHK_SERVICE_OFFSET + 1))
#define  WFS_ERR_CHK_INCOMPLETEREAD             (-(CHK_SERVICE_OFFSET + 2))
#define  WFS_ERR_CHK_FIELDNOTFOUND              (-(CHK_SERVICE_OFFSET + 3))
#define  WFS_ERR_CHK_EXTRAFIELD                 (-(CHK_SERVICE_OFFSET + 4))
#define  WFS_ERR_CHK_FIXEDOVERWRITE             (-(CHK_SERVICE_OFFSET + 5))
#define  WFS_ERR_CHK_UNSUPPORTEDCAP             (-(CHK_SERVICE_OFFSET + 6))
#define  WFS_ERR_CHK_FIELDSPECFAILURE           (-(CHK_SERVICE_OFFSET + 7))
#define  WFS_ERR_CHK_INVALIDHDOC                (-(CHK_SERVICE_OFFSET + 8))
#define  WFS_ERR_CHK_IMAGENOTAVAIL              (-(CHK_SERVICE_OFFSET + 9))
#define  WFS_ERR_CHK_INVALIDCOMMAND             (-(CHK_SERVICE_OFFSET + 10))

/*================================================================*/
```

```
/* CHK Info Command Structures */
/*=====================================================================*/

typedef struct _wfs_chk_status
{
        WORD                    fwDevice;
        WORD                    fwMedia;
        WORD                    fwInk;
        DWORD                   dwMode;
        WORD                    fwLamp;
        LPSTR                   lpszExtra;
} WFSCHKSTATUS, * LPWFSCHKSTATUS;

typedef  struct    _wfs_chk_caps
{
        WORD                    wClass;
        WORD                    fwType;
        BOOL                    bCompound;
        BOOL                    fMICR;
        BOOL                    fOCR;
        BOOL                    fAutoFeed;
        BOOL                    fEndorser;
        BOOL                    fEncoder;
        WORD                    fwStamp;
        WORD                    wImageCapture;
        USHORT                  nPockets;
        LPSTR                   lpszFontNames;
        LPSTR                   lpszEncodeNames;
        LPSTR                   lpszExtra;
} WFSCHKCAPS, * LPWFSCHKCAPS;

typedef struct      _wfs_chk_query_field
{
        LPSTR                   lpszFormName;
        LPSTR                   lpszFieldName;
} WFSCHKQUERYFIELD, * LPWFSCHKQUERYFIELD;

/*=====================================================================*/
/* CHK Execute Command Structures */
/*=====================================================================*/

typedef struct _wfs_chk_in_read_form
{
        LPSTR                   lpszFormName;
        LPSTR                   lpszFieldNames;
        DWORD                   dwOptions;
        LPSTR                   lpszExtra;
} WFSCHKINREADFORM, * LPWFSCHKINREADFORM;

typedef struct _wfs_chk_out_read_form
{
        WORD                    hDoc;
        LPSTR                   lpszFields;
} WFSCHKOUTREADFORM, * LPWFSCHKOUTREADFORM;

typedef struct _wfs_chk_multicommand
```

```
{
        WORD            hDoc;
        DWORD           dwOptions;
        BYTE            bPocket;
        LPSTR           lpszEncodeFormName;
        LPSTR           lpszEncodeFields;
        LPSTR           lpszEndorserFormName;
        LPSTR           lpszEndorserFields;
        LPSTR           lpszExtra;
} WFSCHKMULTICOMMAND, * LPWFSCHKMULTICOMMAND;

typedef struct _wfs_chk_in_read_image
{
        WORD            hDoc;
        DWORD           dwOptions;
        LPSTR           lpszExtra;
} WFSCHKINREADIMAGE, * LPWFSCHKINREADIMAGE;

typedef struct _wfs_chk_out_read_image
{
        WORD            wImage;
        LPSTR           lpImage;
} WFSCHKOUTREADIMAGE, * LPWFSCHKOUTREADIMAGE;


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

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

#endif    /* __INC_XFSCHK__H */
```