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

WORKSHOP AGREEMENT

CWA 14050-7

November 2000

ICS 35.200; 35.240.40

Extensions for Financial Services (XFS) interface specification - Release 3.0 - Part 7: Check Reader/Scanner Device Class Interface

Ref. No CWA 14050-7:2000 E

Rue de Stassart, 36 • B-1050 Bruxelles
Tel : +32 2 550 08 11 • Fax : +32 2 550 08 19

Table of Contents

Foreword

This CWA is revision 3.0 of the XFS interface specification.

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

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

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

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

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

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

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

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

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

Part 2: Service Classes Definition; Programmer's Reference

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Revision History:

1.0	May 24, 1993	Initial release of API and SPI specification
1.01	June 11, 1993	Minor updates to BSVC member contact list.
1.1	April 14, 1994	Major updates and additions.
1.11	February 3, 1995	Separation of specification into separate documents for API/SPI and service class definitions; with updates
3.00	October 18, 2000	Update release encompassing: - Reintroduced with a command set targeted at stand alone check readers and scanners – UNICODE support

1. Introduction

1.1 Background to Release 3.0

The CEN XFS Workshop is a continuation of the Banking Solution Vendors Council workshop and maintains a technical commitment to the Win 32 API. However, the XFS Workshop has extended the franchise of multi vendor software by encouraging the participation of both banks and vendors to take part in the deliberations of the creation of an industry standard. This move towards opening the participation beyond the BSVC's original membership has been very succesful with a current membership level of more than 20 companies.

The fundamental aims of the XFS Workshop are to promote a clear and unambiguous specification for both service providers and application developers. This has been achieved to date by sub groups working electronically and quarterly meetings.

The move from an XFS 2.0 specification to a 3.0 specification has been prompted by a series of factors. Initially, there has been a technical imperative to extend the scope of the existing specification of the XFS Manager to include new devices, such as the Card Embossing Unit.

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

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

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

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 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.

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 desktop units which automatically feed the check one at a time; recording the MICR data and check image, and endorse or encode the check. 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.

This service class is currently defined only for attended branch service

3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer's Reference Revision 3.00, October 18, 2000

4. Info Commands

4.1 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;
  LPSTR         lpszExtra;
  } WFSCHKSTATUS, * LPWFSCHKSTATUS;
```

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

Value	Meaning
WFS_CHK_DEVONLINE	The device is online (i.e., powered on and operable).
WFS_CHK_DEVOFFLINE	The device is offline (e.g., the operator has taken the device offline by turning a switch or pulling out the device).
WFS_CHK_DEVPOWEROFF	The device is powered off or physically not connected.
WFS_CHK_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_CHK_DEVHWERROR	The device is inoperable due to a hardware error.
WFS_CHK_DEVUSERERROR	The device is inoperable because a person is preventing proper device operation.
WFS_CHK_DEVBUSY	The device is busy and unable to process an execute command at this time.

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

Value	Meaning
WFS_CHK_MEDIANOTSUPP	The capability to report the state of the check media is not supported by the device.
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_INKNOTSUPP	Capability not supported by the device.
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.

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

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

4.2 WFS_INF_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;
  LPSTR     lpszFontNames;
  LPSTR     lpszEncodeNames;
  WORD      fwCharSupport;
  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 can read OCR on checks.

bAutoFeed
TRUE if the device has autofeed capability; FALSE if only manual feed.

bEndorser
TRUE if a programmable endorser is present.

bEncoder
TRUE if an encoder is present.

fwStamp
Specifies the physical dimensions of the check where the endorser stamp can be used. A single value can be returned.

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

wImageCapture
Specifies the physical dimensions that can be image captured. A single value can be returned.

Value	Meaning
WFS_CHK_ICAPNONE	Device cannot capture image.
WFS_CHK_ICAPFRONT	Device can image capture front of check.
WFS_CHK_ICAPBACK	Device can image capture back of check.
WFS_CHK_ICAPBOTH	Device can image capture both sides of the check.

lpszFontNames
The names of the fonts supported for reading; each is terminated with a NULL and the string is terminated with two NULLs. Reserved font names include CMC7 and E13B.

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

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

Value	Meaning
WFS_CHK_ASCII	ASCII is supported for XFS forms.
WFS_CHK_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
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 Only the generic error codes defined in [Ref. 1] can be 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.

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

4.4 WFS_INF_CHK_MEDIA_LIST

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

Input Param None.

Output Param `LPSTR         lpszMediaList;`

lpszMediaList
Pointer to a list of null-terminated media 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.

4.5 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

```
LPWFSCHKFRMHEADER   lpFormHeader;

typedef struct _wfs_chk_frm_header
   {
   LPSTR     lpszFormName;
   WORD      wBase;
   WORD      wUnitX;
   WORD      wUnitY;
   WORD      wWidth;
   WORD      wHeight;
   WORD      wAlignment;
   WORD      wOffsetX;
   WORD      wOffsetY;
   WORD      wVersionMajor;
   WORD      wVersionMinor;
   WORD      fwCharSupport;
   LPSTR     lpszFields;
   } WFSCHKFRMHEADER, * LPWFSCHKFRMHEADER;
```

lpszFormName
Specifies the null-terminated name of the form.

wBase
Specifies the base unit of measurement of the form and can be one of the following:

Value	Meaning
WFS_CHK_INCH	The base unit is inches.
WFS_CHK_MM	The base unit is millimeters.
WFS_CHK_ROWCOLUMN	The base unit is rows and columns.

wUnitX
Specifies the horizontal resolution of the base units as a fraction of the *wBase* value. For example, a value of 16 applied to the base unit WFS_CHK_INCH means that the base horizontal resolution is 1/16".

wUnitY
Specifies the vertical resolution of the base units as a fraction of the *wBase* value. For example, a value of 10 applied to the base unit WFS_CHK_MM means that the base vertical resolution is 0.1 mm.

wWidth
Specifies the width of the form in terms of the base horizontal resolution.

wHeight
Specifies the height of the form in terms of the base vertical resolution.

wAlignment
Specifies the relative alignment of the form on the media and can be one of the following:

Value	Meaning
WFS_CHK_TOPLEFT	The form is aligned relative to the top and left edges of the media.
WFS_CHK_TOPRIGHT	The form is aligned relative to the top and right edges of the media.
WFS_CHK_BOTTOMLEFT	The form is aligned relative to the bottom and left edges of the media.
WFS_CHK_BOTTOMRIGHT	The form is aligned relative to the bottom and right edges of the media.

wOffsetX
Specifies the horizontal offset of the position of the top-left corner of the form, relative to the left or right edge specified by *wAlignment*. This value is specified in terms of the base horizontal resolution and is always positive.

wOffsetY
Specifies the vertical offset of the position of the top-left corner of the form, relative to the top or bottom edge specified by *wAlignment*. This value is specified in terms of the base vertical resolution and is always positive.

wVersionMajor
Specifies the major version of the form.

wVersionMinor
Specifies the minor version of the form.

fwCharSupport
A single flag specifying the Character Set in which the form is encoded:

Value	Meaning
WFS_CHK_ASCII	ASCII is supported for XFS forms initial data values and FORMAT strings.
WFS_CHK_UNICODE	UNICODE is supported for XFS forms initial data values and FORMAT strings.

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

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

Value	Meaning
WFS_ERR_CHK_FORMNOTFOUND	The specified form cannot be found.
WFS_ERR_CHK_FORMINVALID	The specified form is invalid.

4.6 WFS_INF_CHK_QUERY_MEDIA

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

Input Param `LPSTR         lpszMediaName;`

lpszMediaName
Pointer to the null-terminated media name about which to retrieve details.

Output Param `LPWFSCHKFRMMEDIA    lpFormMedia;`

```
typedef struct _wfs_chk_frm_media
    {
    WORD      fwMediaType;
    WORD      wBase;
    WORD      wUnitX;
    WORD      wUnitY;
    WORD      wSizeWidth;
    WORD      wSizeHeight;
    WORD      wCheckAreaX;
    WORD      wCheckAreaY;
    WORD      wCheckAreaWidth;
    WORD      wCheckAreaHeight;
    WORD      wRestrictedAreaX;
    WORD      wRestrictedAreaY;
    WORD      wRestrictedAreaWidth;
    WORD      wRestrictedAreaHeight;
    } WFSCHKFRMMEDIA, * LPWFSCHKFRMMEDIA;
```

fwMediaType
Specifies the type of media as one of the following flags:

Value	Meaning
WFS_CHK_MEDIACHECK	Check media.

wBase
Specifies the base unit of measurement of the form and can be one of the following:

Value	Meaning
WFS_CHK_INCH	The base unit is inches.
WFS_CHK_MM	The base unit is millimeters.
WFS_CHK_ROWCOLUMN	The base unit is rows and columns.

wUnitX
Specifies the horizontal resolution of the base units as a fraction of the *wBase* value. For example, a value of 16 applied to the base unit WFS_CHK_INCH means that the base horizontal resolution is 1/16".

wUnitY
Specifies the vertical resolution of the base units as a fraction of the *wBase* value. For example, a value of 10 applied to the base unit WFS_CHK_MM means that the base vertical resolution is 0.1 mm.

wSizeWidth
Specifies the width of the media in terms of the base horizontal resolution.

wSizeHeight
Specifies the height of the media in terms of the base vertical resolution.

wCheckAreaX
Specifies the horizontal offset of the Check area relative to the top left corner of the media in terms of the base horizontal resolution.

wCheckAreaY
Specifies the vertical offset of the Check area relative to the top left corner of the media in terms of the base vertical resolution.

wCheckAreaWidth
Specifies the Check area width of the media in terms of the base horizontal resolution.

wCheckAreaHeight
Specifies the Check area height of the media in terms of the base vertical resolution.

wRestrictedAreaX
Specifies the horizontal offset of the restricted area relative to the top left corner of the media in terms of the base horizontal resolution.

wRestrictedAreaY
Specifies the vertical offset of the restricted area relative to the top left corner of the media in terms of the base vertical resolution.

wRestrictedAreaWidth
Specifies the restricted area width of the media in terms of the base horizontal resolution.

wRestrictedAreaHeight
Specifies the restricted area height of the media in terms of the base vertical resolution.

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_CHK_MEDIANOTFOUND	The specified media definition cannot be found.
WFS_ERR_CHK_MEDIAINVALID	The specified media definition is invalid.

Comments None.

4.7 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 lpQueryField;

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

lpszFormName
Points to the null-terminated form name.

lpszFieldName
Pointer to the null-terminated name of the field about which to retrieve details.
If the value of *lpszFieldName* is a NULL pointer, then details are retrieved for all fields on the form. Depending upon whether the form is encoded in UNICODE representation either the *lpszInitialValue* or *lpszUNICODEInitialValue* output fields are used to retrieve the field Initial Value.

Output Param

```
LPWFSCHKFRMFIELD *  lppFields;
```

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

```
typedef struct _wfs_chk_frm_field
    {
    LPSTR      lpszFieldName;
    WORD       fwType;
    WORD       fwClass;
    WORD       fwAccess;
    WORD       fwOverflow;
    LPSTR      lpszInitialValue;
    LPWSTR     lpszUNICODEInitialValue;
    LPSTR      lpszFormat;
    LPWSTR     lpszUNICODEFormat;
    } WFSCHKFRMFIELD, * LPWFSCHKFRMFIELD;
```

lpszFieldName
Pointer to the null-terminated field name.

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

Value	Meaning
WFS_CHK_FIELDTEXT	A text field.
WFS_CHK_FIELDMICR	A Magnetic Ink Character Recognition (MICR) field.
WFS_CHK_FIELDOCR	An Optical Character Recognition (OCR) field.
WFS_CHK_FIELDGRAPHIC	A Graphic field

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

Value	Meaning
WFS_CHK_CLASSSTATIC	The field data cannot be set by the application.
WFS_CHK_CLASSOPTIONAL	The field data can be set by the application.
WFS_CHK_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_CHK_ACCESSREAD	The field is used for input.
WFS_CHK_ACCESSWRITE	The field is used for output.

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

Value	Meaning
WFS_CHK_OVFTERMINATE	Return an error and terminate printing of the form.
WFS_CHK_OVFTRUNCATE	Truncate the field data to fit in the field.
WFS_CHK_OVFBESTFIT	Fit the text in the field.
WFS_CHK_OVFOVERWRITE	Print the field data beyond the extents of the field boundary.
WFS_CHK_OVFWORDWRAP	If the field can hold more than one line the text is wrapped around.

lpszInitialValue
The initial value of the field when the field is written as output.

lpszUNICODEInitialValue
The initial value of the field when form is encoded in UNICODE.

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

lpszUNICODEFormat
Format string as defined in the form for this field when form is encoded in UNICODE.

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_CHK_FORMNOTFOUND	The specified form cannot be found.
WFS_ERR_CHK_FORMINVALID	The specified form is invalid.
WFS_ERR_CHK_FIELDNOTFOUND	The specified field cannot be found.
WFS_ERR_CHK_FIELDINVALID	The specified field is invalid.
WFS_ERR_CHK_CHARSETDATA	The character set(s) found is not supported by the service provider.

5. Execute Commands

5.1 WFS_CMD_CHK_PROCESS_FORM

Description This function initiates feeding and processing of a check. Based on the form definition and dwOptions field, the MICR/OCR data is read, check image is scanned, check is endorsed, and MICR/OCR is written. Depending upon the check reader/scanner unit, for each WFS_CMD_CHK_PROCESS_FORM command executed, a single feed/eject of the check will usually occur.

If the invoking application needs to read the check MICR/OCR data prior to knowing what to write to the check in the form of endorsement data or MICR/OCR data then a WFS_CMD_CHK_PROCESS_FORM command must first be executed with a null lpszOutputFields field or dwOptions field set to WFS_CHK_NO_WRITE followed by another WFS_CMD_CHK_PROCESS_FORM command with appropriate lpszOutputFields field content to be written.

Input Param LPWFSCHKINPROCESSFORM lpChkInProcessForm;

```
typedef struct _wfs_chk_in_process_form
    {
    LPSTR      lpszFormName;
    LPSTR      lpszMediaName;
    LPSTR      lpszInputFields;
    LPSTR      lpszOutputFields;
    LPWSTR     lpszUNICODEOutputFields;
    DWORD      dwOptions;
    } WFSCHKINPROCESSFORM, * LPWFSCHKINPROCESSFORM;
```

lpszFormName
Points to the null-terminated name of the form

lpszMediaName
Points to the null-terminated media name.

lpszInputFields
Pointer to a list of null-terminated field names from which to read input data, with the final name terminating with two null characters. If *lpszInputFields* contains two null characters then no data is read (no MICR/OCR fields are read).

lpszOutputFields
Pointer to a series of "<FieldName>=<FieldValue>" strings, where each string is null-terminated with the entire field string terminating with two null characters. If *lpszOutputFields* contains two null characters then no data is written (no data is endorsed and no MICR/OCR is written).

lpszUNICODEOutputFields
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.
The *lpszUNICODEOutputFields* field should only be used if the form is encoded in UNICODE representation. This can be determined with the WFS_CHK_INF_QUERY_FORM command.

dwOptions
One or more of the following flags are set.

Value	Meaning
WFS_CHK_OPT_AUTOFEED	Auto feed check (Check automatically feed and ejected)
WFS_CHK_OPT_ICAPFRONT	Image capture (scan image) front of check.
WFS_CHK_OPT_ICAPBACK	Image capture (scan image) back of check.
WFS_CHK_OPT_NO_MICR_OCR	Do not read MICR/OCR fields.
WFS_CHK_OPT_NO_WRITE	Do not write text or graphic output fields.

Output Param `LPWFSCHKOUTPROCESSFORM     lpOutProcessForm;`

```
typedef struct _wfs_chk_out_process_form
   {
   LPSTR    lpszInputFields;
   LPWSTR   lpszUNICODEInputFields;
   WORD     wFrontImageType;
   ULONG    ulFrontImageSize;
   LPBYTE   lpFrontImage;
   WORD     wBackImageType;
   ULONG    ulBackImageSize;
   LPBYTE   lpBackImage;
   } WFSCHKOUTPROCESSFORM, * LPWFSCHKOUTPROCESSFORM;
```

lpszInputFields
Pointer to a series of "<FieldName>=<FieldValue>" strings, where each string is null-terminated with the entire input field string terminating with two null characters.
Contains a sequence such as (given a U.S. personal check):

```
ROUTETRANS=021203501\0ACCOUNT=370361\0TRANCODE=2199\0AMOUNT=000000
1000\0\0
```

lpszUNICODEInputFields
Pointer to a series of "<FieldName>=<FieldValue>" UNICODE strings, where each string is null-terminated with the entire input field string terminating with two null characters.

wFrontImageType
Specifies the format of the front of the check image returned by this command as one of the following flags:

Value	Meaning
WFS_CHK_IMAGETIF	The returned image is in TIF format.
WFS_CHK_IMAGEMTF	The returned image is in MTF format (Metafile format)
WFS_CHK_IMAGEBMP	The returned image is in BMP format.

ulFrontImageSize
Count of bytes of front of check image data.

lpFrontImage
Points to the front of check image data.

wBackImageType
Specifies the format of the back of the check image returned by this command as one of the following flags:

Value	Meaning
WFS_CHK_IMAGETIF	The returned image is in TIF format.
WFS_CHK_IMAGEMTF	The returned image is in MTF format (Metafile format)
WFS_CHK_IMAGEBMP	The returned image is in BMP format.

ulBackImageSize
Count of bytes of back of check image data.

lpBackImage
Points to the back of check image data.

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_CHK_REQDFIELDMISSING	A required field is missing on the check.
WFS_ERR_CHK_FORMNOTFOUND	The specified form cannot be found.
WFS_ERR_CHK_FORMINVALID	The specified form definition is invalid.
WFS_ERR_CHK_MEDIANOTFOUND	The specified media definition cannot be found.
WFS_ERR_CHK_MEDIAINVALID	The specified media definition is invalid.
WFS_ERR_CHK_MEDIAOVERFLOW	The form overflowed the media.
WFS_ERR_CHK_FIELDSPECFAILURE	The syntax of the *lpszInputFields* or *lpszOutputFields* member is invalid.

WFS_ERR_CHK_FIELDERROR	An error occurred while processing a field, causing termination of the read request. An execute event WFS_EXEE_CHK_FIELDERROR is posted with the details.
WFS_ERR_CHK_CHARSETDATA	Character set(s) supported by service provider is inconsistent with use of *lpszOutputField* or *lpszUNICODEOutputField*.
WFS_ERR_CHK_MEDIAJAM	The media is jammed. Operator intervention is required.
WFS_ERR_CHK_SHUTTERFAIL	The device is unable to open and/or close it's shutter.

Events In addition to the generic events defined in [Ref. 1], the following 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.
WFS_EXEE_CHK_FIELDERROR	A fatal error occurred while processing a field.
WFS_EXEE_CHK_FIELDWARNING	A non-fatal error occurred while processing a field.
WFS_USRE_CHK_INKTHRESHOLD	The toner or ink supply is low or empty or the printing contrast with ribbon is weak or not sufficient, operator intervention is required. Note that this event is sent only once, at the point at which the toner becomes low or empty. It is sent with WFS_CHK_INKLOW or WFS_CHK_INKOUT status.

Comments. The timeout parameter (*dwTimeOut*) in the **WFSExecute** request that passes this command should always be large enough to accommodate prompting the operator to insert a check, having the operator do so, and processing the check.

The application will use *lpszOutputField* or *lpszUNICODEOutputField* as an input parameter, depending upon the service provider capabilities. Legacy (non-UNICODE aware) applications will only use the *lpszOutputField* field. UNICODE applications can use either the *lpszOutputField* or *lpszUNICODEOutputField* fields, provided the service provider is UNICODE compliant.

5.2 WFS_CMD_CHK_RESET

Description This command is used by the application to perform a hardware reset which will attempt to return the CHK device to a known good state. This command does not over-ride a lock obtained by another application or service handle.

The device will attempt to either retain, eject or will perform no action on any media found in the CHK as specified in the *lpwResetIn* parameter. It may not always be possible to retain or eject the media as specified because of hardware problems. If a media is found inside the device the WFS_SRVE_CHK_MEDIADETECTED event will inform the application where media was actually moved to. If no action is specified the media will not be moved even if this means that the CHK cannot be recovered.

Input Param `LPWORD  lpwResetIn;`

Specifies the action to be performed on any media found within the CHK as one of the following values :

Value	Meaning
WFS_CHK_RESET_EJECT	Eject any media found.
WFS_CHK_RESET_RETAIN	Retain any media found.

WFS_CHK_RESET_NOACTION	No Action should be performed on any media found.

If this value is a NULL pointer the service provider will determine where to move any media found.

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_CHK_MEDIAJAM	The media is jammed. Operator intervention is required.
WFS_ERR_CHK_SHUTTERFAIL	The device is unable to open and/or close it's shutter.

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

Value	Meaning
WFS_SRVE_CHK_MEDIADETECTED	This event is generated when a media is detected during a reset.

Comments None

6. Events

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.

6.3 WFS_SRVE_CHK_MEDIAINSERTED

Description This event specifies that the physical check media has been inserted into the device without any read execute command being executed. This event is only generated when media is entered in an unsolicited manner.

Event Param None.

Comments None.

6.4 WFS_EXEE_CHK_FIELDERROR

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

Event Param

```
LPWFSCHKFIELDFAIL   lpFieldFail;

typedef struct _wfs_chk_field_failure
   {
   LPSTR        lpszFormName;
   LPSTR        lpszFieldName;
   WORD         wFailure;
   } WFSCHKFIELDFAIL, * LPWFSCHKFIELDFAIL;
```

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_CHK_FIELDREQUIRED	The specified field ***must*** be supplied by the application.
WFS_CHK_FIELDSTATICOVWR	The specified field is static and thus ***cannot*** be overwritten by the application.
WFS_CHK_FIELDOVERFLOW	The value supplied for the specified fields is too long.

WFS_CHK_FIELDNOTFOUND	The specified field does not exist.
WFS_CHK_FIELDNOTREAD	The specified field is not an input field.
WFS_CHK_FIELDNOTWRITE	An attempt was made to write to an input field.
WFS_CHK_FIELDHWERROR	The specified field uses special hardware (e.g., OCR) and an error occurred.
WFS_CHK_FIELDTYPENOTSUPPORTED	The form field type is not supported with device.

6.5 WFS_EXEE_CHK_FIELDWARNING

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

Event Param `LPWFSPTRFIELDFAIL   lpFieldFail;`

as defined in the section describing WFS_EXEE_CHK_FIELDERROR.

Comments None.

6.6 WFS_USRE_CHK_INKTHRESHOLD

Description This user event is used to specify that the state of the ink reached a threshold.

Event Param `LPWORD lpwInkThreshold;`

Specified as one of the following flags:

Value	Meaning
WFS_CHK_INKFULL	The ink is in a good state.
WFS_CHK_INKLOW	The ink is low.
WFS_CHK_INKOUT	The ink is out.

Comments None.

6.7 WFS_SRVE_CHK_MEDIADETECTED

Description This service event is generated if media is detected during a reset (WFS_CMD_CHK_RESET). The parameter on the event informs the application of the position of the media on the completion of the reset.

Event Param `LPWORD lpwResetOut;`

Specifies the position of any media found within the CHK as one of the following values:

Value	Meaning
WFS_CHK_MEDIAEJECTED	The media was ejected.
WFS_CHK_MEDIARETAINED	The media was retained.
WFS_CHK_MEDIAJAMMED	The media is jammed in the device.

7. Forms Language Usage

This section covers the usage of the forms language to accommodate check readers.

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. For forms intended for use with check readers, the FORMAT attribute is required. The FORMAT keyword is application defined, however may be interpreted by the service provider. The following illustrates the use of the FORMAT keyword:

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

Field names are not limited to the sample field names above. Punctuation marks are used 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.).

Fields are processed in the order that they appear within the Form definition. 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.

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:

- The values of a keyword are separated by commas.
- 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.
- All forms can be represented using either ISO 646 (ANSI) or UNICODE character encoding. If the UNICODE representation is used then all Names and Strings are restricted to an internal representation of ISO 646 (ANSI) characters. Only the INITIALVALUE and FORMAT keyword values can have double byte values outside of the ISO 646 (ANSI) character set.

- If forms character encoding is UNICODE then, consistent with the UNICODE standard, the file prefix must be in little endian (xFFFE) or big endian (xFEFF) notation, such that UNICODE encoding is recognized.

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 dimensions of checks are not identical
- Just-in-time form loading may not be supported by all Service Providers, which makes it impossible to create dynamic form files just before scanning
- Some form/media definition keywords may not be supported due to limitations of the hardware or software

7.3 Form and Media Measurements

The UNIT keyword sections of the form and media definitions specify the base horizontal and vertical resolution as follows:

- the *base* value specifies the base unit of measurement
- the x and y values specify the horizontal and vertical resolution as fractions of the base value (e.g., an *x* value of 10 and a base value of MM means that the base horizontal resolution is 0.1mm).

The base resolutions thus defined by the UNIT keyword section of the ***form*** definition are used as the units of the form definition keyword sections:

- SIZE (*width* and *height* values)
- ALIGNMENT (*xoffset* and *yoffset* values)

and of the field definition keyword sections:

- POSITION (*x* and *y* values)
- SIZE (*width* and *height* values)

The base resolutions thus defined by the UNIT keyword section of the ***media*** definition are used as the units of the media definition keyword sections:

- SIZE (*width* and *height* values)
- CHECKAREA (*x*, *y*, *width* and *height* values)
- RESTRICTED (*x*, *y*, *width* and *height* values)

7.4 Form Definition

XFSFORM		*formname*	
BEGIN			
(required)	**UNIT**	*base,*	Base resolution unit for form definition MM INCH ROWCOLUMN
		x,	Horizontal base unit fraction
		y	Vertical base unit fraction
(required)	**SIZE**	width,	Width of form
		height	Height of form
	ALIGNMENT	alignment,	Alignment of the form on the physical medium: TOPLEFT (default) TOPRIGHT BOTTOMLEFT BOTTOMRIGHT
		xoffset,	Horizontal offset relative to the horizontal alignment specified by alignment. Always specified as a positive value (i.e., if aligned to the right side of the medium, means offset the form to the left). (default = 0)
		yoffset	Vertical offset relative to the vertical alignment specified by alignment. Always specified as a positive value (i.e., if aligned to the bottom of the medium, means offset the form upward). (default = 0)
	VERSION	*major,*	Major version number
		minor,	Minor version number
		date,*	Creation/modification date
		*author**	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
	USERPROMPT	*prompt**	Prompt string for user interaction
	[XFSFIELD	*fieldname*	One field definition (as defined in the next section) for each field in the form

	BEGIN
	. . .
	END]		

END			

7.5 Field Definition

XFSFIELD		*fieldname*	
BEGIN			
(required)	**POSITION**	*x,* *y*	Horizontal position (relative to left or right side of form, depending upon HPOSITION keyword) Vertical position (relative to top or bottom of form, depending upon VPOSITION keyword)
	HPOSITION		Horizontal field positioning relative to: LEFT (default) RIGHT
	VPOSITION		Vertical field positioning relative to: TOP BOTTOM (default)
	TYPE	*fieldtype*	Type of field: GRAPHIC MICR (default) OCR TEXT
	LANGUAGE	languageID	Language used in this field – 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) If unspecified defaults to form definition LANGUAGE specification.
	SIDE		Side of check. FRONT (default) BACK
	CLASS	*class*	Field class OPTIONAL (default) STATIC REQUIRED
	ACCESS	access	Access rights of field WRITE (default) READ
	OVERFLOW	overflow	Action on field overflow: TERMINATE (default) TRUNCATE BESTFIT (the service provider fits the data into the field as well as it can) OVERWRITE (a contiguous write) WORDWRAP
	CASE	*case*	Convert field contents to NOCHANGE (default) UPPER LOWER
	HORIZONTAL	*justify*	Horizontal alignment of field contents LEFT (default) RIGHT CENTER JUSTIFY

	VERTICAL	*justify*	Vertical alignment of field contents BOTTOM (default) CENTER TOP
(required)	**SIZE**	*width,* *height*	Field width Field height
	STYLE	style	Display attributes as a combination of the following, ORed together using the "|" operator: NORMAL (default) BOLD ITALIC UNDER (single underline) DOUBLEUNDER (double underline) DOUBLE (double width) TRIPLE (triple width) QUADRUPLE (quadruple width) STRIKETHROUGH ROTATE90 (rotate +90 degrees clockwise) ROTATE270 (rotate +270 degrees clockwise) UPSIDEDOWN (upside down) PROPORTIONAL (proportional spacing) DOUBLEHIGH TRIPLEHIGH QUADRUPLEHIGH CONDENSED SUPERSCRIPT SUBSCRIPT OVERSCORE LETTERQUALITY NEARLETTERQUALITY DOUBLESTRIKE OPAQUE (If omitted then default attribute is transparent) Some of these Styles may be mutually exclusive, or may combine to provide unexpected results.
	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 GRAPHIC field types.
	FONT	*fontname**	For **MICR or OCR WRITE** field, in some cases this predefines the following parameters: CMC7 E13B For **TEXT** field, font name is interpreted by the service provider. In some cases it may indicate printer resident fonts, and in others it may indicate the name of a downloadable font.
Definition	**POINTSIZE**	pointsize	Point size
Information	**CPI**	cpi	Characters per inch

	LPI	lpi	Lines per inch
(required)	**FORMAT**	*formatstring**	For **MICR or OCR READ** field, the *formatstring* is used to generate the delimiters for the check fields; its usage is application defined. The FORMAT keyword may also be interpreted by the service provider. To have the MICR/OCR check line fields parsed, the field names must be defined. The FORMAT keyword for three fields are illustrated as follows: Amount FORMAT “;NNNNNNNNNN;” AccountNum FORMAT “0000NNNNNN<” RouteTransit FORMAT “;NNNNNNNNN;” Field names are not limited to the above sample field names. To define the entire MICR/OCR check line as an unparsed field to be returned to the application, a field must be defined with the name “MICROCRDATA”. Punctuation marks are used in place of the standard field separators. A capital N means a number is 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.). For **TEXT** field, 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 may 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). Graphic file name may be full or partial path. For example “C:\XFS\XFSLOGO.BMP” illustrates use of full path name. A file name specification of “LOGO.BMP” illustrates partial path name. In this instance file is obtained from current directory.
END			

7.6 Media Definition

The media definition determines those characteristics that result from the combination of a particular media type together with a particular check. The aim is to make it easy to move forms between different checks which might have different constraints on how they handle a specific media type. It is the service provider's responsibility to ensure that the form definition does not specify the reading/writing of any fields that conflict with the media definition. An example of such a conflict might be that the form definition asks for a field to be read/written in an area that the media definition defines as a restricted area.

XFSMEDIA		medianame*	
BEGIN			
	TYPE	type	Predefined media types are: CHECK
(required)	**UNIT**	base, x, y,	Base resolution unit for media definition MM INCH ROWCOLUMN Horizontal base unit fraction Vertical base unit fraction
(required)	**SIZE**	width, height	Width of physical media Height of physical media
	CHECKAREA	x, y, width, height	Check area relative to top left corner of physical media (default = physical size of media)
	RESTRICTED	x, y, width, height	Restricted area relative to to top left corner of physical media (default = no restricted area)
END			

8. C - Header file

```
/******************************************************************************
*                                                                            *
* xfschk.h      XFS - Check reader/scanner (CHK) definitions                 *
*                                                                            *
*               Version 3.00 (10/18/00)                                      *
*                                                                            *
******************************************************************************/

#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 WFSCHKCAPS.wClass */

#define    WFS_SERVICE_CLASS_CHK             (5)
#define    WFS_SERVICE_VERSION_CHK           (0x0003) /* Version 3.00 */
#define    WFS_SERVICE_NAME_CHK              "CHK"

#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_MEDIA_LIST            (CHK_SERVICE_OFFSET + 4)
#define    WFS_INF_CHK_QUERY_FORM            (CHK_SERVICE_OFFSET + 5)
#define    WFS_INF_CHK_QUERY_MEDIA           (CHK_SERVICE_OFFSET + 6)
#define    WFS_INF_CHK_QUERY_FIELD           (CHK_SERVICE_OFFSET + 7)

/* CHK Command Verbs */

#define    WFS_CMD_CHK_PROCESS_FORM          (CHK_SERVICE_OFFSET + 1)
#define    WFS_CMD_CHK_RESET                 (CHK_SERVICE_OFFSET + 2)

/* CHK Messages */

#define    WFS_EXEE_CHK_NOMEDIA              (CHK_SERVICE_OFFSET + 1)
#define    WFS_EXEE_CHK_MEDIAINSERTED        (CHK_SERVICE_OFFSET + 2)
#define    WFS_SRVE_CHK_MEDIAINSERTED        (CHK_SERVICE_OFFSET + 3)
#define    WFS_EXEE_CHK_FIELDERROR           (CHK_SERVICE_OFFSET + 4)
#define    WFS_EXEE_CHK_FIELDWARNING         (CHK_SERVICE_OFFSET + 5)
#define    WFS_USRE_CHK_INKTHRESHOLD         (CHK_SERVICE_OFFSET + 6)
#define    WFS_SRVE_CHK_MEDIADETECTED        (CHK_SERVICE_OFFSET + 7)


/* values of WFSCHKSTATUS.fwDevice  */

#define    WFS_CHK_DEVONLINE                 WFS_STAT_DEVONLINE
#define    WFS_CHK_DEVOFFLINE                WFS_STAT_DEVOFFLINE
#define    WFS_CHK_DEVPOWEROFF               WFS_STAT_DEVPOWEROFF
#define    WFS_CHK_DEVNODEVICE               WFS_STAT_DEVNODEVICE
#define    WFS_CHK_DEVUSERERROR              WFS_STAT_DEVUSERERROR
#define    WFS_CHK_DEVHWERROR                WFS_STAT_DEVHWERROR
#define    WFS_CHK_DEVBUSY                   WFS_STAT_DEVBUSY

/* values of WFSCHKSTATUS.fwMedia, WFS_SRVE_CHK_MEDIADETECTED event */

#define    WFS_CHK_MEDIANOTSUPP              (0)
#define    WFS_CHK_MEDIANOTPRESENT           (1)
#define    WFS_CHK_MEDIAREQUIRED             (2)
```

```
#define    WFS_CHK_MEDIAPRESENT                (3)
#define    WFS_CHK_MEDIAJAMMED                 (4)
#define    WFS_CHK_MEDIAEJECTED                (5)
#define    WFS_CHK_MEDIARETAINED               (6)

/* values of WFSCHKSTATUS.fwInk, lpwInkThreshold */

/* values of WFSCHKCAPS.fwType */
#define    WFS_CHK_TYPECHK                     (1)

#define    WFS_CHK_INKNOTSUPP                  (0)
#define    WFS_CHK_INKFULL                     (1)
#define    WFS_CHK_INKLOW                      (2)
#define    WFS_CHK_INKOUT                      (3)

/* values of WFSCHKCAPS.fwStamp */

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

/* values of WFSCHKCAPS.wImageCapture */

#define    WFS_CHK_ICAPNONE                    (1)
#define    WFS_CHK_ICAPFRONT                   (2)
#define    WFS_CHK_ICAPBACK                    (3)
#define    WFS_CHK_ICAPBOTH                    (4)

/* values of WFSCHKCAPS.fwCharSupport, WFSCHKFRMHEADER.fwCharSupport  */

#define    WFS_CHK_ASCII                       (0x0001)
#define    WFS_CHK_UNICODE                     (0x0002)

/* values of WFSCHKFRMHEADER.wBase, WFSCHKFRMMEDIA.wBase  */

#define    WFS_CHK_INCH                        (1)
#define    WFS_CHK_MM                          (2)
#define    WFS_CHK_ROWCOLUMN                   (3)

/* values of WFSCHKFRMHEADER.wAlignment */

#define    WFS_CHK_TOPLEFT                     (1)
#define    WFS_CHK_TOPRIGHT                    (2)
#define    WFS_CHK_BOTTOMLEFT                  (3)
#define    WFS_CHK_BOTTOMRIGHT                 (4)

/* values of WFSCHKFRMMEDIA.fwMediaType  */

#define    WFS_CHK_MEDIACHECK                  (1)

/* values of WFSCHKFRMFIELD.fwType  */

#define    WFS_CHK_FIELDTEXT                   (1)
#define    WFS_CHK_FIELDMICR                   (2)
#define    WFS_CHK_FIELDOCR                    (3)
#define    WFS_CHK_FIELDGRAPHIC                (4)

/* values of WFSCHKFRMFIELD.fwClass  */

#define    WFS_CHK_CLASSSTATIC                 (1)
#define    WFS_CHK_CLASSOPTIONAL               (2)
#define    WFS_CHK_CLASSREQUIRED               (3)

/* values of WFSCHKFRMFIELD.fwAccess  */

#define    WFS_CHK_ACCESSREAD                  (1)
#define    WFS_CHK_ACCESSWRITE                 (2)
```

```
/* values of WFSCHKFRMFIELD.fwOverflow */

#define    WFS_CHK_OVFTERMINATE              (0)
#define    WFS_CHK_OVFTRUNCATE               (1)
#define    WFS_CHK_OVFBESTFIT                (2)
#define    WFS_CHK_OVFOVERWRITE              (3)
#define    WFS_CHK_OVFWORDWRAP               (4)

/* values of WFSCHKINPROCESSFORM.dwOptions  */

#define    WFS_CHK_OPT_AUTOFEED              0x0001
#define    WFS_CHK_OPT_ICAPFRONT             0x0002
#define    WFS_CHK_OPT_ICAPBACK              0x0004
#define    WFS_CHK_OPT_NO_MICR_OCR           0x0008
#define    WFS_CHK_OPT_NO_WRITE              0x0010

/* values of WFSCHKOUTPROCESSFORM.wFrontImageType, WFSCHKOUTPROCESSFORM.wBackImageType
*/

#define    WFS_CHK_IMAGETIF                  (1)
#define    WFS_CHK_IMAGEMTF                  (2)
#define    WFS_CHK_IMAGEBMP                  (3)

/* input values to WFS_CMD_CHK_RESET */

#define    WFS_CHK_RESET_EJECT               (1)
#define    WFS_CHK_RESET_CAPTURE             (2)
#define    WFS_CHK_RESET_NOACTION            (3)


/* CHK Errors */

#define    WFS_ERR_CHK_FORMNOTFOUND          (-(CHK_SERVICE_OFFSET + 0))
#define    WFS_ERR_CHK_FORMINVALID           (-(CHK_SERVICE_OFFSET + 1))
#define    WFS_ERR_CHK_MEDIANOTFOUND         (-(CHK_SERVICE_OFFSET + 2))
#define    WFS_ERR_CHK_MEDIAINVALID          (-(CHK_SERVICE_OFFSET + 3))
#define    WFS_ERR_CHK_MEDIAOVERFLOW         (-(CHK_SERVICE_OFFSET + 4))
#define    WFS_ERR_CHK_FIELDNOTFOUND         (-(CHK_SERVICE_OFFSET + 5))
#define    WFS_ERR_CHK_FIELDINVALID          (-(CHK_SERVICE_OFFSET + 6))
#define    WFS_ERR_CHK_FIELDERROR            (-(CHK_SERVICE_OFFSET + 7))
#define    WFS_ERR_CHK_REQDFIELDMISSING      (-(CHK_SERVICE_OFFSET + 8))
#define    WFS_ERR_CHK_FIELDSPECFAILURE      (-(CHK_SERVICE_OFFSET + 9))
#define    WFS_ERR_CHK_CHARSETDATA           (-(CHK_SERVICE_OFFSET + 10))
#define    WFS_ERR_CHK_MEDIAJAM              (-(CHK_SERVICE_OFFSET + 11))
#define    WFS_ERR_CHK_SHUTTERFAIL           (-(CHK_SERVICE_OFFSET + 12))

/* values of WFSCHKFIELDFAIL.wFailure  */

#define    WFS_CHK_FIELDREQUIRED             (1)
#define    WFS_CHK_FIELDSTATICOVWR           (2)
#define    WFS_CHK_FIELDOVERFLOW             (3)
#define    WFS_CHK_FIELDNOTFOUND             (4)
#define    WFS_CHK_FIELDNOTREAD              (5)
#define    WFS_CHK_FIELDNOTWRITE             (6)
#define    WFS_CHK_FIELDHWERROR              (7)
#define    WFS_CHK_FIELDTYPENOTSUPPORTED     (8)


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

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

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;
    LPSTR     lpszFontNames;
    LPSTR     lpszEncodeNames;
    WORD      fwCharSupport;
    LPSTR     lpszExtra;
} WFSCHKCAPS, * LPWFSCHKCAPS;

typedef struct _wfs_chk_frm_header
{
    LPSTR     lpszFormName;
    WORD      wBase;
    WORD      wUnitX;
    WORD      wUnitY;
    WORD      wWidth;
    WORD      wHeight;
    WORD      wAlignment;
    WORD      wOffsetX;
    WORD      wOffsetY;
    WORD      wVersionMajor;
    WORD      wVersionMinor;
    WORD      fwCharSupport;
    LPSTR     lpszFields;
} WFSCHKFRMHEADER, * LPWFSCHKFRMHEADER;

typedef struct _wfs_chk_frm_media
{
    WORD      fwMediaType;
    WORD      wBase;
    WORD      wUnitX;
    WORD      wUnitY;
    WORD      wSizeWidth;
    WORD      wSizeHeight;
    WORD      wCheckAreaX;
    WORD      wCheckAreaY;
    WORD      wCheckAreaWidth;
    WORD      wCheckAreaHeight;
    WORD      wRestrictedAreaX;
    WORD      wRestrictedAreaY;
    WORD      wRestrictedAreaWidth;
    WORD      wRestrictedAreaHeight;
} WFSCHKFRMMEDIA, * LPWFSCHKFRMMEDIA;

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

typedef struct _wfs_chk_frm_field
{
    LPSTR     lpszFieldName;
    WORD      fwType;
    WORD      fwClass;
    WORD      fwAccess;
    WORD      fwOverflow;
    LPSTR     lpszInitialValue;
    LPWSTR    lpszUNICODEInitialValue;
    LPSTR     lpszFormat;
    LPWSTR    lpszUNICODEFormat;
} WFSCHKFRMFIELD, * LPWFSCHKFRMFIELD;

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

```
typedef struct _wfs_chk_in_process_form
{
    LPSTR   lpszFormName;
    LPSTR   lpszMediaName;
    LPSTR   lpszInputFields;
    LPSTR   lpszOutputFields;
    LPWSTR  lpszUNICODEOutputFields;
    DWORD   dwOptions;
} WFSCHKINPROCESSFORM, * LPWFSCHKINPROCESSFORM;

typedef struct _wfs_chk_out_process_form
{
    LPSTR   lpszInputFields;
    LPWSTR  lpszUNICODEInputFields;
    WORD    wFrontImageType;
    ULONG   ulFrontImageSize;
    LPBYTE  lpFrontImage;
    WORD    wBackImageType;
    ULONG   ulBackImageSize;
    LPBYTE  lpBackImage;
} WFSCHKOUTPROCESSFORM, * LPWFSCHKOUTPROCESSFORM;

typedef struct _wfs_chk_field_failure
{
    LPSTR   lpszFormName;
    LPSTR   lpszFieldName;
    WORD    wFailure;
} WFSCHKFIELDFAIL, * LPWFSCHKFIELDFAIL;

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

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

#endif /* __INC_XFSCHK__H */
```