Extensions for Financial Services (XFS) interface specification
Release 3.20 - Part 17: Barcode Reader Device Class Interface
Programmer’s Reference

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-CENELEC Management Centre 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, Bulgaria, Croatia, Cyprus, Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, Netherlands, Norway, Poland, Portugal, Romania, Slovakia, Slovenia, Spain, Sweden, Switzerland and United Kingdom.
# Table of Contents

FOREWORD ....................................................................................................................... 3

1. Introduction ................................................................................................................ 6
   1.1 Background to Release 3.20 .................................................................................... 6
   1.2 XFS Service-Specific Programming ....................................................................... 6

2. Barcode Readers ..................................................................................................... 7

3. References .............................................................................................................. 8

4. Info Commands ....................................................................................................... 9
   4.1 WFS_INF_BCR_STATUS .......................................................................................... 9
   4.2 WFS_INF_BCR_CAPABILITIES .......................................................................... 12

5. Execute Commands .............................................................................................. 15
   5.1 WFS_CMD_BCR_READ .......................................................................................... 15
   5.2 WFS_CMD_BCR_RESET ...................................................................................... 17
   5.3 WFS_CMD_BCR_SET_GUIDANCE_LIGHT ............................................................ 18
   5.4 WFS_CMD_BCR_POWER_SAVE_CONTROL ......................................................... 19

6. Events .................................................................................................................... 20
   6.1 WFS_SRVE_BCR DEVICEPOSITION ..................................................................... 20
   6.2 WFS_SRVE_BCR_POWER_SAVE_CHANGE .......................................................... 21

7. C - Header file ....................................................................................................... 22
Foreword

This CWA is revision 3.20 of the XFS interface specification.

This CEN Workshop Agreement has been drafted and approved by a Workshop of representatives of interested parties on 2011-06-29, the constitution of which was supported by CEN following the public call for participation made on 1998-06-24. The specification is continuously reviewed and commented in the CEN/ISSS Workshop on XFS. It is therefore expected that an update of the specification will be published in due time as a CWA, superseding this revision 3.20.

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

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

Part 1: Application Programming Interface (API) - Service Provider Interface (SPI) - Programmer's Reference
Part 2: Service Classes Definition - Programmer's Reference
Part 3: Printer and Scanning Device Class Interface - Programmer's Reference
Part 4: Identification Card Device Class Interface - Programmer's Reference
Part 5: Cash Dispenser Device Class Interface - Programmer's Reference
Part 6: PIN Keypad Device Class Interface - Programmer's Reference
Part 7: Check Reader/Scanner Device Class Interface - Programmer's Reference
Part 8: Depository Device Class Interface - Programmer's Reference
Part 9: Text Terminal Unit Device Class Interface - Programmer's Reference
Part 10: Sensors and Indicators Unit Device Class Interface - Programmer's Reference
Part 11: Vendor Dependent Mode Device Class Interface - Programmer's Reference
Part 12: Camera Device Class Interface - Programmer's Reference
Part 13: Alarm Device Class Interface - Programmer's Reference
Part 14: Card Embossing Unit Class Interface - Programmer's Reference
Part 15: Cash-In Module Device Class Interface - Programmer's Reference
Part 16: Card Dispenser Device Class Interface - Programmer's Reference
Part 17: Barcode Reader Device Class Interface - Programmer's Reference
Part 18: Item Processing Module Device Class Interface - Programmer's Reference
Parts 19 - 28: Reserved for future use.

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

Part 29: XFS MIB Architecture and SNMP Extensions
Part 30: XFS MIB Device Specific Definitions - Printer Device Class
Part 31: XFS MIB Device Specific Definitions - Identification Card Device Class
Part 32: XFS MIB Device Specific Definitions - Cash Dispenser Device Class
Part 33: XFS MIB Device Specific Definitions - PIN Keypad Device Class
Part 34: XFS MIB Device Specific Definitions - Check Reader/Scanner Device Class
Part 35: XFS MIB Device Specific Definitions - Depository Device Class
Part 36: XFS MIB Device Specific Definitions - Text Terminal Unit Device Class
Part 37: XFS MIB Device Specific Definitions - Sensors and Indicators Unit Device Class
Part 38: XFS MIB Device Specific Definitions - Camera Device Class
Part 39: XFS MIB Device Specific Definitions - Alarm Device Class
Part 40: XFS MIB Device Specific Definitions - Card Embossing Unit Device Class
Part 41: XFS MIB Device Specific Definitions - Cash-In Module Device Class
Part 42: Reserved for future use.
Part 43: XFS MIB Device Specific Definitions - Vendor Dependent Mode Class
Part 44: XFS MIB Application Management
Part 45: XFS MIB Device Specific Definitions - Card Dispenser Device Class
Part 46: XFS MIB Device Specific Definitions - Barcode Reader Device Class
Part 47: XFS MIB Device Specific Definitions - Item Processing Module Device Class
Parts 48 - 60 are reserved for future use.
Part 61: Application Programming Interface (API) - Service Provider Interface (SPI) - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 62: Printer and Scanning Device Class Interface - Migration from Version 3.10 (CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 63: Identification Card Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 64: Cash Dispenser Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 65: PIN Keypad Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 66: Check Reader/Scanner Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 67: Depository Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 68: Text Terminal Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 69: Sensors and Indicators Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 70: Vendor Dependent Mode Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 71: Camera Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 72: Alarm Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 73: Card Embossing Unit Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 74: Cash-In Module Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 75: Card Dispenser Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 76: Barcode Reader Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
Part 77: Item Processing Module Device Class Interface - Migration from Version 3.10 (see CWA 15748) to Version 3.20 (this CWA) - Programmer's Reference
In addition to these Programmer's Reference specifications, the reader of this CWA is also referred to a

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

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

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

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

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

Revision History:

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.10</td>
<td>November 29, 2007</td>
<td>Initial release.</td>
</tr>
<tr>
<td>3.20</td>
<td>March 2nd, 2011</td>
<td>For a description of changes from version 3.10 to version 3.20 see the BCR 3.20 Migration document.</td>
</tr>
</tbody>
</table>
1. Introduction

1.1 Background to Release 3.20

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

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

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

1.2 XFS Service-Specific Programming

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

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

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

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

1. 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.

2. 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.

3. 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. Barcode Readers

This specification describes the functionality of a Barcode Reader (BCR) Service Provider. It defines the service-specific commands that can be issued to the Service Provider using the WFSGetInfo, WFSAsyncGetInfo, WFSExecute and WFSAsyncExecute functions.

Persistent values are maintained through power failures, open sessions, close session and system resets.

This extension to XFS specifications defines the functionality of BCR service.

A Barcode Reader scans barcodes using any scanning technology. The device logic converts light signals or image recognition into application data and transmits it to the host system.

The basic operation of the Barcode Reader is managed using WFSExecute/WFSAsyncExecute functions.

When an application wants to read a barcode, it issues a WFS_CMD_BCR_READ command to prepare the scanner to read any barcode presented to it. When a document is presented to the BCR and a barcode type is recognized, a completion event is received which contains the barcode data that has been read.
3. References

1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer’s Reference Revision 3.20
4. Info Commands

4.1 WFS_INF_BCR_STATUS

Description
This command is used to request status information for the device.

Input Param
None.

Output Param
LPWFSBCRSTATUS lpStatus;

typedef struct _wfs_bcr_status
{
    WORD    fwDevice;
    WORD    fwBCRScanner;
    DWORD    dwGuidLights[WFS_BCR_GUIDLIGHTS_SIZE];
    LPSTR    lpszExtra;
    WORD    wDevicePosition;
    USHORT   usPowerSaveRecoveryTime;
    WORD    wAntiFraudModule;
} WFSBCRSTATUS, *LPWFSBCRSTATUS;

fwDevice
Specifies the state of the BCR device as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_BCR_DEVONLINE</td>
<td>The device is online (i.e. powered on and operable).</td>
</tr>
<tr>
<td>WFS_BCR_DEVOFFLINE</td>
<td>The device is offline (e.g. the operator has taken the device offline by turning a switch or pulling out the device).</td>
</tr>
<tr>
<td>WFS_BCR_DEVPOWEROFF</td>
<td>The device is powered off or physically not connected.</td>
</tr>
<tr>
<td>WFS_BCR_DEVNODEVICE</td>
<td>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.</td>
</tr>
<tr>
<td>WFS_BCR_DEVHWERROR</td>
<td>The device is inoperable due to a hardware error.</td>
</tr>
<tr>
<td>WFS_BCR_DEVUSERERROR</td>
<td>The device is present but a person is preventing proper device operation.</td>
</tr>
<tr>
<td>WFS_BCR_DEVBUSY</td>
<td>The device is busy and unable to process an execute command at this time.</td>
</tr>
<tr>
<td>WFS_BCR_DEVFRAUDATTEMPT</td>
<td>The device is present but is inoperable because it has detected a fraud attempt.</td>
</tr>
<tr>
<td>WFS_BCR_DEVPOTENTIALFRAUD</td>
<td>The device has detected a potential fraud attempt and is capable of remaining in service. In this case the application should make the decision as to whether to take the device offline.</td>
</tr>
</tbody>
</table>

fwBCRScanner
Specifies the scanner status (laser, camera or other technology) as one of the following flags:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_BCR_SCANNERON</td>
<td>Scanner is enabled for reading.</td>
</tr>
<tr>
<td>WFS_BCR_SCANNEROFF</td>
<td>Scanner is disabled.</td>
</tr>
<tr>
<td>WFS_BCR_SCANNERINOP</td>
<td>Scanner is inoperative due to a hardware error.</td>
</tr>
<tr>
<td>WFS_BCR_SCANNERUNKNOWN</td>
<td>Scanner status cannot be determined.</td>
</tr>
</tbody>
</table>

dwGuidLights [...] summons all guidance light indicators. A number of guidance light types are defined below. Vendor specific guidance lights are defined starting from the end of the array. The maximum guidance light index is WFS_BCR_GUIDLIGHTS_MAX.
Specifies the state of the guidance light indicator as WFS_BCR_GUIDANCE_NOT_AVAILABLE, WFS_BCR_GUIDANCE_OFF or a combination of the following flags consisting of one type B, and optionally one type C.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_BCR_GUIDANCE_NOT_AVAILABLE</td>
<td>The status is not available.</td>
<td>A</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_OFF</td>
<td>The light is turned off.</td>
<td>A</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_SLOW_FLASH</td>
<td>The light is blinking slowly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_MEDIUM_FLASH</td>
<td>The light is blinking medium frequency.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_QUICK_FLASH</td>
<td>The light is blinking quickly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_CONTINUOUS</td>
<td>The light is turned on continuous</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_RED</td>
<td>The light is red.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_GREEN</td>
<td>The light is green.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_YELLOW</td>
<td>The light is yellow.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_BLUE</td>
<td>The light is blue.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_CYAN</td>
<td>The light is cyan.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_MAGENTA</td>
<td>The light is magenta.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_WHITE</td>
<td>The light is white.</td>
<td>C</td>
</tr>
</tbody>
</table>

dwGuidLights [WFS_BCR_GUIDANCE_BCR]
Specifies the state of the guidance light indicator on the Barcode Reader unit.

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

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_BCR_DEVICEINPOSITION</td>
<td>The device is in its normal operating position, or is fixed in place and cannot be moved.</td>
</tr>
<tr>
<td>WFS_BCR_DEVICENOTINPOSITION</td>
<td>The device has been removed from its normal operating position.</td>
</tr>
<tr>
<td>WFS_BCR_DEVICEPOSUNKNOWN</td>
<td>Due to a hardware error or other condition, the position of the device cannot be determined.</td>
</tr>
<tr>
<td>WFS_BCR_DEVICEPOSNOTSUPP</td>
<td>The physical device does not have the capability of detecting the position.</td>
</tr>
</tbody>
</table>

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

wAnti Fraud Module
Specifies the state of the anti-fraud module as one of the following values:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_BCR_AFMNOTSUPP</td>
<td>No anti-fraud module is available.</td>
</tr>
<tr>
<td>WFS_BCR_AFMOK</td>
<td>Anti-fraud module is in a good state and no foreign device is detected.</td>
</tr>
<tr>
<td>WFS_BCR_AFINOP</td>
<td>Anti-fraud module is inoperable.</td>
</tr>
<tr>
<td>WFS_BCR_AFMDEVICEDETECTED</td>
<td>Anti-fraud module detected the presence of a foreign device.</td>
</tr>
</tbody>
</table>
WFS_BCR_AFMUNKNOWN The state of the anti-fraud module cannot be determined.

**Error Codes**

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

**Comments**

In the case where communications with the device has been lost, the *fwDevice* field will report WFS_BCR_DEVPOWEROFF when the device has been removed or WFS_BCR_DEVHWERROR if the communications are unexpectedly lost. All other fields should contain a value based on the following rules and priority:

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

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

Input Param
None.

Output Param
LPWFSBCRCAPS lpCaps;

typedef struct _wfs_bcr_caps
{
    WORD wClass;
    BOOL bCompound;
    BOOL bCanFilterSymbologies;
    LPWORD lpwSymbologies;
    DWORD dwGuidLights[WFS_BCR_GUIDLIGHTS_SIZE];
    LPSTR lpszExtra;
    BOOL bPowerSaveControl;
    BOOL bAntiFraudModule;
} WFSBCRCAPS, *LPWFSBCRCAPS;

wClass
Specifies the logical service class as WFS_SERVICE_CLASS_BCR.

bCompound
Specifies whether the logical device is part of a compound physical device.

bCanFilterSymbologies
Specifies whether the device is capable of discriminating between the presented barcode symbologies such that only the desired symbologies are recognized/reported.

lpwSymbologies
Pointer to an array of WORDs. This list specifies the barcode symbologies readable by the scanner. The array is terminated with a zero value. lpwSymbologies is a NULL pointer if the supported barcode symbologies can not be determined.

Value | Meaning
--- | ---
WFS_BCR_SYM_EAN128 | GS1-128
WFS_BCR_SYM_EAN8 | EAN-8
WFS_BCR_SYM_EAN8_2 | EAN-8 with 2 digit add-on
WFS_BCR_SYM_EAN8_5 | EAN-8 with 5 digit add-on
WFS_BCR_SYM_EAN13 | EAN13
WFS_BCR_SYM_EAN13_2 | EAN-13 with 2 digit add-on
WFS_BCR_SYM_EAN13_5 | EAN-13 with 5 digit add-on
WFS_BCR_SYM_JAN13 | JAN-13
WFS_BCR_SYM_UPCA | UPC-A
WFS_BCR_SYM_UPCE0 | UPC-E
WFS_BCR_SYM_UPCE0_2 | UPC-E with 2 digit add-on
WFS_BCR_SYM_UPCE0_5 | UPC-E with 5 digit add-on
WFS_BCR_SYM_UPCE1 | UPC-E with leading 1
WFS_BCR_SYM_UPCE1_2 | UPC-E with leading 1 and 2 digit add-on
WFS_BCR_SYM_UPCE1_5 | UPC-E with leading 1 and 5 digit add-on
WFS_BCR_SYM_UPCA_2 | UPC-A with 2 digit add-on
WFS_BCR_SYM_UPCA_5 | UPC-A with 5 digit add-on
WFS_BCR_SYM_CODABAR | CODABAR (NW-7)
WFS_BCR_SYM_ITF | Interleaved 2 of 5 (ITF)
WFS_BCR_SYM_11 | CODE 11 (USD-8)
WFS_BCR_SYM_39 | CODE 39
WFS_BCR_SYM_49 | CODE 49
WFS_BCR_SYM_93 | CODE 93
WFS_BCR_SYM_128 | CODE 128
WFS_BCR_SYM_MSI | MSI
WFS_BCR_SYM_PLESSEY | PLESSEY
WFS_BCR_SYM_STD2OF5 | STANDARD 2 of 5 (INDUSTRIAL 2 of 5 also)
WFS_BCR_SYM_STD2OF5_IATA | STANDARD 2 of 5 (IATA Version)
WFS_BCR_SYM_PDF_417 | PDF-417
Specifies which guidance lights are available. A number of guidance light types are defined below. Vendor specific guidance lights are defined starting from the end of the array. The maximum guidance light index is WFS_BCR_GUIDLIGHTS_MAX.

The elements of this array are specified as a combination of the following flags and indicate all of the possible flash rates (type B) and colors (type C) that the guidance light indicator is capable of handling. If the guidance light indicator only supports one color then no value of type C is returned. A value of WFS_BCR_GUIDANCE_NOT_AVAILABLE indicates that the device has no guidance light indicator or the device controls the light directly with no application control possible.

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_BCR_GUIDANCE_NOT_AVAILABLE</td>
<td>There is no guidance light control available at this position.</td>
<td>A</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_OFF</td>
<td>The light can be off.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_SLOW_FLASH</td>
<td>The light can blink slowly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_MEDIUM_FLASH</td>
<td>The light can blink medium frequency.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_QUICK_FLASH</td>
<td>The light can blink quickly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_CONTINUOUS</td>
<td>The light can be continuous (steady).</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_RED</td>
<td>The light can be red.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_GREEN</td>
<td>The light can be green.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_YELLOW</td>
<td>The light can be yellow.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_BLUE</td>
<td>The light can be blue.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_CYAN</td>
<td>The light can be cyan.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_MAGENTA</td>
<td>The light can be magenta.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_WHITE</td>
<td>The light can be white.</td>
<td>C</td>
</tr>
</tbody>
</table>

dwGuidLights [WFS_BCR_GUIDANCE_BCR]

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

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

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

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

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

5.1 WFS_CMD_BCR_READ

Description
This command enables the barcode reader. The barcode reader will scan for barcodes and when it successfully manages to read one or more barcodes the command will complete. The completion event for this command contains the scanned barcode data.

Input Param
LPWFSBCRREADINPUT lpReadInput;

```c
typedef struct _wfs_bcr_read_input {
    LPWORD   lpwSymbologies;
} WFSBCRREADINPUT, *LPWFSBCRREADINPUT;
```

lpwSymbologies
Array specifying a list that contains the sub-set of bar code symbologies that the application wants to be accepted for this command. The array is terminated with a zero value. In some cases the Service Provider can discriminate between barcode symbologies and return the data only if the presented symbology matches with one of the desired symbologies. See the bCanFilterSymbologies capability to determine if the Service Provider supports this feature. If the Service Provider does not support this feature then this parameter is ignored. If all symbologies should be accepted then lpwSymbologies should be set to NULL.

Output Param
LPWFSBCRREADOUTPUT *lppReadOutput;

Pointer to a NULL terminated array of pointers to WFSBCRREADOUTPUT structures. There is one array element for each barcode read during the scan.

```c
typedef struct _wfs_bcr_read_output {
    WORD    wSymbology;
    LPWFSBCRXDATA  lpxBarcodeData;
    LPSTR    lpszSymbologyName;
} WFSBCRREADOUTPUT, *LPWFSBCRREADOUTPUT;
```

wSymbology
Specifies the barcode symbology recognized. This contains one of the values returned in the lpwSymbologies field of the WFS_INF_BCR_CAPABILITIES command. If the barcode reader is unable to recognize the symbology as one of the values reported via the device capabilities then the value for this field will be WFS_BCR_SYM_UNKNOWN.

lpxBarcodeData
Contains the barcode data read from the barcode reader. The format of the data will depend on the barcode symbology read. In most cases this will be an array of bytes containing ASCII numeric digits. However, the format of the data in this field depends entirely on the symbology read, e.g. it may contain 8 bit character values where the symbol is dependent on the codepage used to encode the barcode, may contain UNICODE data, or may be a binary block of data. The application is responsible for checking the completeness and validity of the data.

lpszSymbologyName
A vendor dependent symbology identifier for the symbology recognized.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_BCR_BARCODEINVALID</td>
<td>The read operation could not be completed successfully. The barcode presented was defective or was wrongly read.</td>
</tr>
</tbody>
</table>

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

Comments
The device waits for the period of time specified by the dwTimeOut parameter in the WFSEexecute call for one of the enabled symbologies to be presented, unless the hardware has a fixed timeout period that is less than the value passed in the WFSEexecute command.

The data type LPWFSBCRXDATA is used to return the barcode data.
typedef struct _wfs_bcr_hex_data
{
    USHORT    usLength;
    LPBYTE    lpbData;
} WFSBCRXDATA, *LPWFSBCRXDATA;

usLength
Length of the byte stream pointed to by lpbData.

lpbData
Pointer to the data stream.
### 5.2 WFS_CMD_BCR_RESET

<table>
<thead>
<tr>
<th>Description</th>
<th>This command is used to reset the device. The scanner returns to power-on initial status and remains disabled for any barcode label reading.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input Param</td>
<td>None.</td>
</tr>
<tr>
<td>Output Param</td>
<td>None.</td>
</tr>
<tr>
<td>Error Codes</td>
<td>Only the generic errors codes defined in [Ref. 1] can be generated by this command.</td>
</tr>
<tr>
<td>Events</td>
<td>Only the generic events defined in [Ref. 1] can be generated by this command.</td>
</tr>
<tr>
<td>Comments</td>
<td>None.</td>
</tr>
</tbody>
</table>
5.3 WFS_CMD_BCR_SET_GUIDANCE_LIGHT

Description
This command is used to set the status of the BCR guidance lights. This includes defining the flash rate and the color. When an application tries to use a color that is not supported then the Service Provider will return the generic error WFS_ERR_UNSUPP_DATA.

Input Param
LPWFSBCRSETGUIDLIGHT lpSetGuidLight;

define struct _wfs_bcr_set_guidlight
{
    WORD    wGuidLight;
    DWORD    dwCommand;
} WFSBCRSETGUIDLIGHT, *LPWFSBCRSETGUIDLIGHT;

wGuidLight
Specifies the index of the guidance light to set as one of the values defined within the capabilities section.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_BCR_GUIDANCE_OFF</td>
<td>The light indicator is turned off.</td>
<td>A</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_SLOW_FLASH</td>
<td>The light indicator is set to flash slowly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_MEDIUM_FLASH</td>
<td>The light indicator is set to flash medium frequency.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_QUICK_FLASH</td>
<td>The light indicator is set to flash quickly.</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_CONTINUOUS</td>
<td>The light indicator is turned on continuously (steady).</td>
<td>B</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_RED</td>
<td>The light indicator color is set to red.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_GREEN</td>
<td>The light indicator color is set to green.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_YELLOW</td>
<td>The light indicator color is set to yellow.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_BLUE</td>
<td>The light indicator color is set to blue.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_CYAN</td>
<td>The light indicator color is set to cyan.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_MAGENTA</td>
<td>The light indicator color is set to magenta.</td>
<td>C</td>
</tr>
<tr>
<td>WFS_BCR_GUIDANCE_WHITE</td>
<td>The light indicator color is set to white.</td>
<td>C</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_BCR_INVALID_PORT</td>
<td>An attempt to set a guidance light to a new value was invalid because the guidance light does not exist.</td>
</tr>
</tbody>
</table>

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

Comments
The slow and medium flash rates must not be greater than 2.0 Hz. It should be noted that in order to comply with American Disabilities Act guidelines only a slow or medium flash rate must be used.
5.4 WFS_CMD_BCR_POWER_SAVE_CONTROL

**Description**  This command activates or deactivates the power-saving mode.

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

**Input Param**  
LPWFSBCRPOWERSAVECONTROL lpPowerSaveControl;

typedef struct _wfs_bcr_power_save_control {
    USHORT usMaxPowerSaveRecoveryTime;
} WFSBCRPOWERSAVECONTROL, *LPWFSBCRPOWERSAVECONTROL;

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

**Output Param**  None.

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_ERR_BCR_POWERSAVETOOSHORT</td>
<td>The power saving mode has not been activated because the device is not able to resume from the power saving mode within the specified <em>usMaxPowerSaveRecoveryTime</em> value.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_SRVE_BCR_POWER_SAVE_CHANGE</td>
<td>The power save recovery time has changed.</td>
</tr>
</tbody>
</table>

**Comments**  None.
6. Events

6.1 WFS_SRVE_BCR_DEVICEPOSITION

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

Event Param
LPWFSBCRDEVICEPOSITION lpDevicePosition;

typedef struct _wfs_bcr_device_position
{
    WORD wPosition;
} WFSBCRDEVICEPOSITION, *LPWFSBCRDEVICEPOSITION;

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

<table>
<thead>
<tr>
<th>Value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFS_BCR_DEVICEINPOSITION</td>
<td>The device is in its normal operating position.</td>
</tr>
<tr>
<td>WFS_BCR_DEVICENOTINPOSITION</td>
<td>The device has been removed from its normal operating position.</td>
</tr>
<tr>
<td>WFS_BCR_DEVICEPOSUNKNOWN</td>
<td>The position of the device cannot be determined.</td>
</tr>
</tbody>
</table>

Comments
None.
### 6.2 WFS_SRVE_BCR_POWER_SAVE_CHANGE

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

**Event Param**

```c
typedef struct _wfs_bcr_power_save_change
{
    USHORT   usPowerSaveRecoveryTime;
} WFSBCRPOWERSAVECHANGE, *LPWFSBCRPOWERSAVECHANGE;
```

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

**Comments**: If another device class compounded with this device enters into a power saving mode this device will automatically enter into the same power saving mode and this event will be generated.
7. C - Header file

/*******************************************************************************
* xfsbcr.h    XFS - Barcode Reader (BCR) definitions                        *
* Revision 3.20  (March 02 2011)                                           *
******************************************************************************/

#ifndef __INC_XFSBCR__H
#define __INC_XFSBCR__H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

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

/* values of WFSBCRCAPS.wClass */
#define WFS_SERVICE_CLASS_BCR               (15)
#define WFS_SERVICE_CLASS_VERSION_BCR       (0x1403) /* Version 3.20 */
#define WFS_SERVICE_CLASS_NAME_BCR          "BCR"
#define BCR_SERVICE_OFFSET                  (WFS_SERVICE_CLASS_BCR * 100)

/* BCR Info Commands */
#define WFS_INF_BCR_STATUS                  (BCR_SERVICE_OFFSET + 1)
#define WFS_INF_BCR_CAPABILITIES            (BCR_SERVICE_OFFSET + 2)

/* BCR Execute Commands */
#define WFS_CMD_BCR_READ                    (BCR_SERVICE_OFFSET + 1)
#define WFS_CMD_BCR_RESET                   (BCR_SERVICE_OFFSET + 2)
#define WFS_CMD_BCR_SET_GUIDANCE_LIGHT      (BCR_SERVICE_OFFSET + 3)
#define WFS_CMD_BCR_POWER_SAVE_CONTROL      (BCR_SERVICE_OFFSET + 4)

/* BCR Messages */
#define WFS_SRVE_BCR_DEVICEPOSITION         (BCR_SERVICE_OFFSET + 1)
#define WFS_SRVE_BCR_POWER_SAVE_CHANGE      (BCR_SERVICE_OFFSET + 2)

/* values of WFSBCRSTATUS.fwDevice */
#define WFS_BCR_DEVONLINE                   WFS_STAT_DEVONLINE
#define WFS_BCR_DEVOFFLINE                  WFS_STAT_DEVOFFLINE
#define WFS_BCR_DEVPOWEROFF                 WFS_STAT_DEVPOWEROFF
#define WFS_BCR_DEVNODEVICE                 WFS_STAT_DEVNODEVICE
#define WFS_BCR_DEVUSERERROR                WFS_STAT_DEVUSERERROR
#define WFS_BCR_DEVBUSY                     WFS_STAT_DEVBUSY
#define WFS_BCR_DEVFRAUDATTEMPT             WFS_STAT_DEVFRAUDATTEMPT
#define WFS_BCR_DEVPOWEROFF                 WFS_STAT_DEVPOWEROFF
#define WFS_BCR_DEVPOTENTIALFRAUD           WFS_STAT_DEVPOTENTIALFRAUD

/* values of WFSBCRSTATUS.fwBCRScanner */
#define WFS_BCR_SCANNERON                   (0)
#define WFS_BCR_SCANNEROFF                  (1)
#define WFS_BCR_SCANNERINOP                 (2)
#define WFS_BCR_SCANNERUNKNOWN              (3)

/* values of WFSBCRSTATUS.wDevicePosition */
#define WFS_BCR_DEVINPOSITION               (0)
#define WFS_BCR_DEVPOWEROFFOFF              (1)
#define WFS_BCR_DEVPOWERSUPPORT             (2)

#endif
#endif

#include <xfsrcapi.h>
/* values of WFSBRCAPS.lpwSymbologies
   WFSBCRREADINPUT.lpwSymbologies
   WFSBCRREADOUTPUT.wSymbology */

#define     WFS_BCR_SYM_UNKNOWN                 (0)
#define     WFS_BCR_SYM_EAN128                  (1)
#define     WFS_BCR_SYM_EAN8                    (2)
#define     WFS_BCR_SYM_EAN8_2                  (3)
#define     WFS_BCR_SYM_EAN8_5                  (4)
#define     WFS_BCR_SYM_EAN13                   (5)
#define     WFS_BCR_SYM_EAN13_2                 (6)
#define     WFS_BCR_SYM_EAN13_5                 (7)
#define     WFS_BCR_SYM_JAN13                   (8)
#define     WFS_BCR_SYM_UPCA                    (9)
#define     WFS_BCR_SYM_UPCE0                   (10)
#define     WFS_BCR_SYM_UPCE0_2                 (11)
#define     WFS_BCR_SYM_UPCE0_5                 (12)
#define     WFS_BCR_SYM_UPCE1                   (13)
#define     WFS_BCR_SYM_UPCE1_2                 (14)
#define     WFS_BCR_SYM_UPCE1_5                 (15)
#define     WFS_BCR_SYM_UPCA_2                  (16)
#define     WFS_BCR_SYM_UPCA_5                  (17)
#define     WFS_BCR_SYM_CODABAR                 (18)
#define     WFS_BCR_SYM_ITF                     (19)
#define     WFS_BCR_SYM_11                      (20)
#define     WFS_BCR_SYM_39                      (21)
#define     WFS_BCR_SYM_49                      (22)
#define     WFS_BCR_SYM_93                      (23)
#define     WFS_BCR_SYM_128                     (24)
#define     WFS_BCR_SYM_MSI                     (25)
#define     WFS_BCR_SYM_PLESSEY                 (26)
#define     WFS_BCR_SYM_STD2OF5                 (27)
#define     WFS_BCR_SYM_STD2OF5_IATA            (28)
#define     WFS_BCR_SYM_PDF_417                 (29)
#define     WFS_BCR_SYM_MICRORF417              (30)
#define     WFS_BCR_SYM_DATAMATRIX              (31)
#define     WFS_BCR_SYM_MAXICODE                (32)
#define     WFS_BCR_SYM_CODEONE                 (33)
#define     WFS_BCR_SYM_CHANNELCODE             (34)
#define     WFS_BCR_SYM_TELEPENEORIG            (35)
#define     WFS_BCR_SYM_TELEPEN_AIM             (36)
#define     WFS_BCR_SYM_TELEPEN                 (37)
#define     WFS_BCR_SYM_RSS                     (38)
#define     WFS_BCR_SYM_RSS_EXPANDED            (39)
#define     WFS_BCR_SYM_RSS_RESTRICTED          (40)
#define     WFS_BCR_SYM_COMPOSITE_CODE_A        (41)
#define     WFS_BCR_SYM_COMPOSITE_CODE_B        (42)
#define     WFS_BCR_SYM_COMPOSITE_CODE_C        (43)
#define     WFS_BCR_SYM_POSICODE_A              (44)
#define     WFS_BCR_SYM_POSICODE_B              (45)
#define     WFS_BCR_SYM_TRIOPTIC_CODE_39        (46)
#define     WFS_BCR_SYM_QRCODE_2                (47)
#define     WFS_BCR_SYM_QRCODE                  (48)
#define     WFS_BCR_SYM_AZTEC                   (49)
#define     WFS_BCR_SYM_UPKPOST                 (50)
#define     WFS_BCR_SYM_PLANET                  (51)
#define     WFS_BCR_SYM_POSTNET                 (52)
#define     WFS_BCR_SYM_CANADIANPOST            (53)
#define     WFS_BCR_SYM_NETHERLANDSPOST         (54)
#define     WFS_BCR_SYM_AUSTRALIANPOST          (55)
#define     WFS_BCR_SYM_JAPANESEPOST            (56)
#define     WFS_BCR_SYM_CHINESEPOST             (57)
#define     WFS_BCR_SYM_KOREANPOST              (58)

/* Size and max index of dwGuidLights array */

#define WFS_BCR_GUIDLIGHTS_SIZE                (32)
#define WFS_BCR_GUIDLIGHTS_MAX                  (WFS_BCR_GUIDLIGHTS_SIZE - 1)

/* Indices of WFSBRCRSTATUS.dwGuidLights [...] 
   WFSBRCRCAPS.dwGuidLights [...] */

#define     WFS_BCR_GUIDANCE_BCR                (0)
/* Values of WFSBCRSTATUS.dwGuidLights [...]
WFSBCRCAPS.dwGuidLights [...],
WFSBCRSETGUIDLIGHT.wGuidLight */

#define     WFS_BCR_GUIDANCE_NOT_AVAILABLE      (0x00000000)
#define     WFS_BCR_GUIDANCE_OFF                (0x00000001)
#define     WFS_BCR_GUIDANCE_ON                 (0x00000002)
#define     WFS_BCR_GUIDANCE_SLOW_FLASH         (0x00000004)
#define     WFS_BCR_GUIDANCE_MEDIUM_FLASH       (0x00000008)
#define     WFS_BCR_GUIDANCE_QUICK_FLASH        (0x00000010)
#define     WFS_BCR_GUIDANCE_CONTINUOUS         (0x00000080)
#define     WFS_BCR_GUIDANCE_RED                (0x00000100)
#define     WFS_BCR_GUIDANCE_GREEN              (0x00000200)
#define     WFS_BCR_GUIDANCE_YELLOW             (0x00000400)
#define     WFS_BCR_GUIDANCE_BLUE               (0x00000800)
#define     WFS_BCR_GUIDANCE_CYAN               (0x00001000)
#define     WFS_BCR_GUIDANCE_MAGENTA            (0x00002000)
#define     WFS_BCR_GUIDANCE_WHITE              (0x00004000)

/* values of WFSBCRSTATUS.wAntiFraudModule */

#define     WFS_BCR_AFMNOTSUPP                  (0)
#define     WFS_BCR_AFMOK                       (1)
#define     WFS_BCR_AFMINOP                     (2)
#define     WFS_BCR_AFMDEVICEDETECTED           (3)
#define     WFS_BCR_AFMUNKNOWN                  (4)

/* XFS BCR Errors */

#define WFS_ERR_BCR_BARCODEINVALID              (~(BCR_SERVICE_OFFSET + 0))
#define WFS_ERR_BCR_INVALID_PORT                (~(BCR_SERVICE_OFFSET + 1))
#define WFS_ERR_BCR_POWERSAVETOOSHORT           (~(BCR_SERVICE_OFFSET + 2))

/*=================================================================*
/* BCR Info Command Structures */
/*=================================================================* /

typedef struct _wfs_bcr_status
{
    WORD             fwDevice;
    WORD             fwBCRScanner;
    DWORD            dwGuidLights[WFS_BCR_GUIDLIGHTS_SIZE];
    LPSTR            lpszExtra;
    WORD             wDevicePosition;
    USHORT           usPowerSaveRecoveryTime;
    WORD             wAntiFraudModule;
} WFSBCRSTATUS, *LPWFSBCRSTATUS;

typedef struct _wfs_bcr_caps
{
    WORD             wClass;
    BOOL             bCompound;
    BOOL             bCanFilterSymbologies;
    LPWORD           lpwSymbologies;
    DWORD            dwGuidLights[WFS_BCR_GUIDLIGHTS_SIZE];
    LPSTR            lpszExtra;
    BOOL             bPowerSaveControl;
    BOOL             bAntiFraudModule;
} WFSBCRCAPS, *LPWFSBCRCAPS;

/*=================================================================*
/* BCR Execute Command Structures */
/*=================================================================* /

typedef struct _wfs_bcr_hex_data
{
    USHORT           usLength;
    LPBYTE           lpbData;
} WFSBCRXDATA, *LPWFSBCRXDATA;

typedef struct _wfs_bcr_read_input
{
    LPWORD           lpwSymbologies;
} WFSBCRREADINPUT, *LPWFSBCRREADINPUT;
typedef struct _wfs_bcr_read_output
{
    WORD             wSymbology;
    LPWFSBCRXDATA   lpxBarcodeData;
    LPSTR            lpszSymbologyName;
} WFSBCRREADOUTPUT, *LPWFSBCRREADOUTPUT;

typedef struct _wfs_bcr_set_guidlight
{
    WORD             wGuidLight;
    DWORD            dwCommand;
} WFSBCRSETGUIDLIGHT, *LPWFSBCRSETGUIDLIGHT;

typedef struct _wfs_bcr_power_save_control
{
    USHORT           usMaxPowerSaveRecoveryTime;
} WFSBCRPOWERSAVECONTROL, *LPWFSBCRPOWERSAVECONTROL;

/*=================================================================* /
/* BCR Message Structures */
/*=================================================================* /

typedef struct _wfs_bcr_device_position
{
    WORD             wPosition;
} WFSBCRDEVICEPOSITION, *LPWFSBCRDEVICEPOSITION;

typedef struct _wfs_bcr_power_save_change
{
    USHORT           usPowerSaveRecoveryTime;
} WFSBCRPOWERSAVECHANGE, *LPWFSBCRPOWERSAVECHANGE;

/*   restore alignment   */
#pragma pack(pop)
#ifdef __cplusplus
} /*extern "C"*/
#endif /* __INC_XFSBCR__H */
#endif /* __INC_XFSBCR__H */