CEN

WORKSHOP

AGREEMENT

CWA 14050-29

July 2007

ICS 35.240.40; 35.200; 35.240.15

English version

Extensions for Financial Services (XFS) interface specification - Release 3.03 - Part 29: XFS MIB Architecture and SNMP Extensions MIB Version 1.1

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 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, 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.

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

Management Centre: rue de Stassart, 36 B-1050 Brussels

Ref. No.:CWA 14050-29:2007 D/E/F

Table of Contents

Foreword

This CWA is revision 3.03 of the XFS interface specification.

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 2004-09-24. 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 3.03.

This document supersedes CWA 14050-29:2004.

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 3.00 (see CWA 14050-4:2000; superseded) - 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 3.00 (see CWA 14050-6:2000; superseded) - 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 3.01 (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

Part 26: Identification Card Device Class Interface - Migration from Version 3.0 (see CWA 14050-4:2000; superseded) to Version 3.02 (this CWA) - Programmer's Reference

Part 27: PIN Keypad Device Class Interface - Migration from Version 3.0 (see CWA 14050-6:2000; superseded) to Version 3.02 (this CWA) - Programmer's Reference

Part 28: Cash In Module Device Class Interface - Migration from Version 3.0 (see CWA 14050-15:2000; superseded) to Version 3.02 (this CWA) - Programmer's Reference

Part 42: PIN Keypad Device Class Interface - Migration from Version 3.02 (see CWA 14050-6:2003; superseded) to Version 3.03 (this CWA) - Programmer's Reference

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

The following parts 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 – Programmer's Reference

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 Class

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

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

Part 44: XFS MIB Application Management

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	20 January 2004	Initial release of XFS MIB specification
1.1	15 April 2007	Updated to add support for Detailed Status Traps, Threshold Traps, VDM MIB, Application Management MIB, Device Reset and support of SMIv2.

This CEN Workshop Agreement is publicly available as a reference document from the National Members of CEN : AENOR, AFNOR, ASRO, BDS, BSI, CSNI, CYS, DIN, DS, ELOT, EVS, IBN, IPQ, IST, LVS, LST, MSA, MSZT, NEN, NSAI, ON, PKN, SEE, SIS, SIST, SFS, SN, SNV, SUTN and UNI

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

1. Introduction

This specification describes the general MIB definition (Management Information Base) for the XFS environment and some new APIs that allow network management of Service Providers from the application layer.
This specification is mainly focused on the following areas:

- SNMP management architecture
- MIB structure definition
- Trap format definition
- Management extension of the Service Providers Interface

Full implementation of the above features depends on the individual vendor-supplied service providers. This specification outlines the functionality and requirements for applications using the XFS network management services, and for the development of those services.

The XFS device specific MIB and the application MIB definitions will be defined in separate documents.

An agent is compliant with the XFS MIB, if it supports the XFS MIB as defined in this specification and the referenced device/application specific XFS MIB specifications. No restrictions are placed on how an agent is implemented.

The MIB feature is an optional addendum to the XFS CWA. In addition, the main focus of this standard is on the standardisation of the MIB specification, not any specific implementation. From a management perspective, the key to multi-vendor management is that the MIB and values are consistent

1.1 Background

This specification defines the integration of the SNMPv1, SNMPv2c and SNMPv3 standards with the set of status and statistical information exported by the XFS Service Providers. This choice allows the adoption of SNMPv1, SNMPv2c and SNMPV3 standards as the Network Management protocol.

All MIB files associated with the XFS MIB Specification comply with SMIv2. MIB files associated with the XFS MIB specification and compliant with SMIv1 are also provided.

Here is a short glossary of terms used in Network Management and specifically about SNMP.

Glossary

IANA Internet Assigned Numbers Authority: it is the authority that assigns Internet addresses and gives the address (OID) to hook to the standard MIB (MIB-II)

MIB Management Information Base: it is the database of information, which SNMP is based on. The database is hierarchical and is managed by the IANA for the assignment of the branch identifier (OID).

OID Object Identifier: it is the identifier of a MIB branch. It identifies unequivocally the architectural content of the MIB (in other words how it is possible to access the information exported by the MIB branch).

SMI Structure of Management Information: The rules used to define the objects that can be accessed via a network management protocol.

SNMP Simple Network Management Protocol: the standard protocol of Network Management. It is based on TCP/IP. The protocol defines two entities involved in communication: SNMP Manager and SNMP Agent.

SNMP Agent The component that responds to requests from the SNMP Manager, either retrieving or setting variables as required. It also generates traps to notify the SNMP Manager when an alarm condition occurs.

SNMP Manager The component that asks the SNMP agents for the information contained in the MIB. It also collects the unsolicited messages coming from the SNMP Agents (traps).

1.2 MIB 1.1 Additions

1.2.1 Threshold Traps and Detailed Status Traps

The XFS SNMP MIB 1.1 specifications have been update to add the following traps:

- A detailed device status change trap that is generated when a device status change (e.g. device goes offline) occurs. This trap is device specific and contains all the information reported via the Device Status Table. The content of the detailed device status change traps is defined within the device MIB specifications.
- A detailed sub-device status change trap that is generated when a sub-device state change occurs (e.g. Cash Unit goes low). This trap is device specific and contains all the information reported via the Sub-Device Status Table. The content of sub-device status change traps is defined within the device MIB specifications.
- A threshold trap that is generated when an XFS USRE event occurs. This trap is device independent and is fully defined within this specification.

These traps have been added because the original traps defined in the XFS SNMP MIB 1.0 specifications do not contain all the detailed information that a remote management system needs. In addition the XFS SNMP MIB 1.0 specifications do not report threshold traps which can be critical for effective dispatch of replenishment/service personnel. The original traps defined in XFS SNMP MIB 1.0 are still valid and are still generated.

1.2.2 SNMP v2c and v3

The XFS SNMP MIB 1.1 specification has been updated to support both SMIv1 and SMIv2. This addition enables SNMPv2c and SNMPv3 to be supported in addition to the existing and ongoing support for SNMPv1.

The XFS MIB 1.1 specifications provide all MIB definitions in both SMIv1 and SMIv2 formats.

1.2.3 Remote Device Reset

The XFS SNMP MIB 1.1 specifications have been updated to allow a management station to attempt a remote Reset on a device in an attempt to bring the device back into service without the need for a field visit. The device specific MIB specifications have been updated with:

- A new Reset Device Table which controls the remote reset request.
- A new Device Reset Completion Trap that reports the result of the Reset request.

This document has been updated with configuration settings to enable/disable the Remote Reset capability and to control the media handling for each device.

1.2.4 VDM Support

The XFS SNMP MIB 1.1 specifications have been updated to support the VDM MIB sub-tree.

1.2.5 Application Management

The XFS SNMP MIB 1.1 specifications have been updated to support summary-level application management features. The details of this feature are defined in the *MIB Application Management* document.

1.3 Architecture

The architecture and information exported for application management is defined within the XFS MIB Application Management specification. The remainder of this specification defines how devices are managed.

The MIB definition specifies what information a Service Provider (i.e. a Service Provider of a generic XFS class) must export in order to be handled by a management application.

The use of information exported by the Service Providers is up to the Solution Providers. They can provide this information to the network management system via SNMP, using an SNMP agent that answers queries on the XFS MIB. They can also use this information for local management.

The exported information is organised into a set of device status variables and a set of response counters. The device status variables describe the current state of the devices (e.g. for a card reader unit, the number of cards retained). The response counters indicate the number of times each response has been returned to each of the execute commands the Service Provider supports.

The management information is presented in logical view, since this is the view presented by XFS. The logical view is provided through the concept of managed services. There is one managed service for every logical interface offered by a physical device. Each managed service has a unique sub-tree within the XFS MIB. Each managed service provides a mapping from the managed services to the physical devices associated with each managed service. This provides support for simple devices with a single interface or compound devices with multiple interfaces. The managed service MIB entries on compound devices are linked through the *xfsManagedServicePhysicalDeviceName* value which contains the same physical device name.

This is best illustrated by way of example:

a) On a standard card reader, there would be one managed service representing the IDC class interface. The MIB would contain a single branch for the managed service. The *xfsManagedServicePhysicalDeviceName* MIB variable would define the name of the physical IDC device.

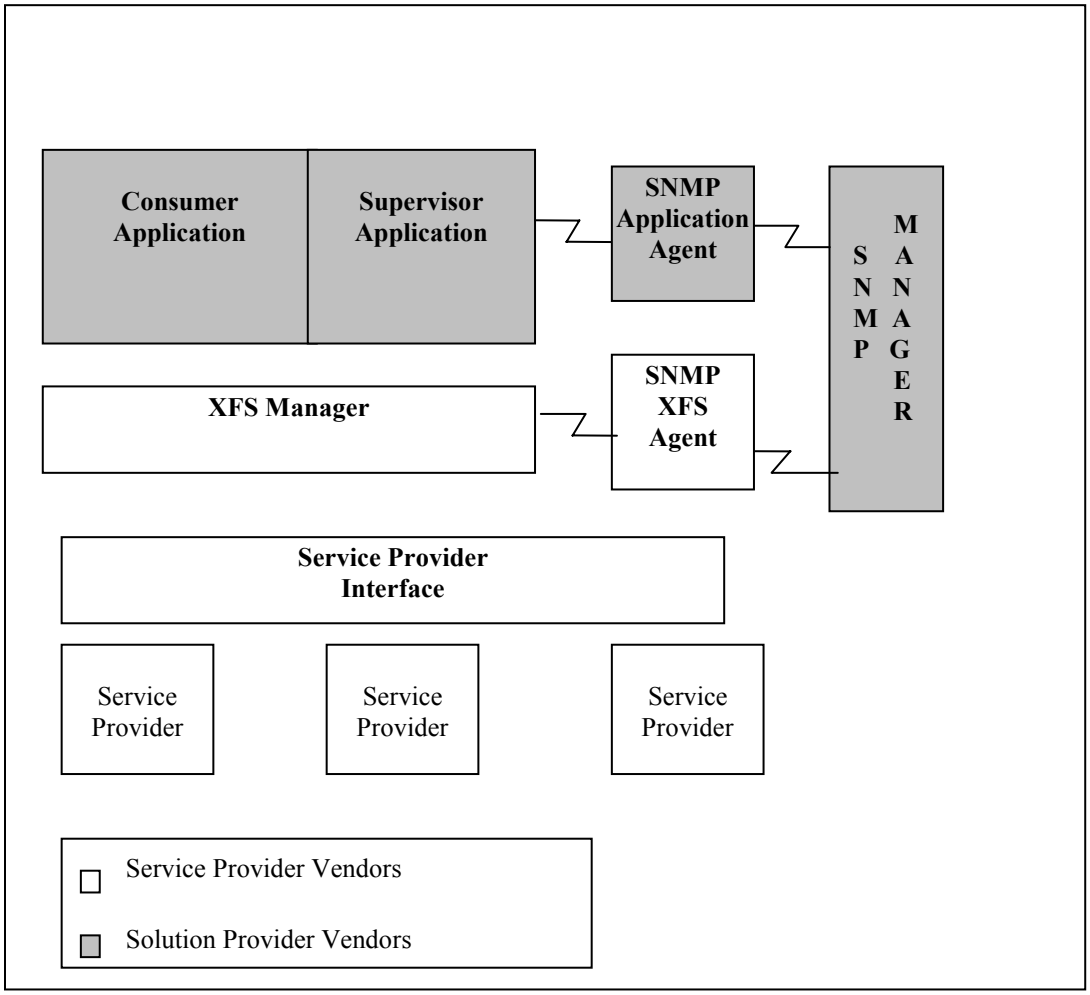
b) On a cash recycler, which is a compound device, there would be one managed service for the CDM interface and one managed service for the CIM interface. The MIB would contain two branches, one for each managed service. The *xfsManagedServicePhysicalDeviceName* MIB variable within the two MIB branches would contain the same name, that of the physical recycle device.

c) On a card reader with a single user card IDC interface and 3 permanent chip IDC interfaces (i.e. 3 permanent chips on the device), there would be 4 managed services in total, one for each logical interface. The MIB would contain four branches, one for each managed service. The *xfsManagedServicePhysicalDeviceName* MIB variable within the four MIB branches would contain the same name, that of the physical IDC device.

The managed services are configured through the XFS registry and are fully described in the “Registry Configuration” section.

Devices, which have logical sub-devices, can also report the status of these sub devices. For example the Cash Units within the CDM and CIM classes are treated as sub-devices.

The solution provider vendors or third parties, using both the SNMP standard interface and the API defined in this document, can develop management applications for the XFS environment.



Management information is divided into general information and device class specific information. General information allows a management application to know the configuration of the installed managed services and associated physical devices. This information is stored in the XFS configuration registry using registry keys as described in the “Registry Configuration” section.

The XFS SNMP agent directly accesses general management information. Device specific information storage is vendor dependent.

The XFS SNMP agent can access device management information through the Service Provider Interface.

A basic feature of the SNMP agent is to be able to notify the remote manager application when an alarm condition occurs (traps). In order to generate traps, the SNMP agent should register for receiving all error and threshold condition notifications from all devices installed on a system. Devices notify error conditions, by the WFS_SYSE_DEVICE_STATUS, the WFS_SYSE_HARDWARE_ERROR and the WFS_SYSE_SOFTWARE_ERROR system events. Devices notify threshold conditions by the WFS_USRE_XXX_THRESHOLD user events. When the agent receives one of the above events, then a trap is generated. On version 1.0 of the MIB only the summary Device Status Change Trap is generated. On version 1.1 of the MIB and higher, both the summary and detailed Device Status change traps will be generated..

2. XFS MIB

The CEN/ISSS XFS Workshop have obtained a Private Enterprise Number from the IANA (Internet Assigned Number Authority). The Private Enterprise Number assigned to the CEN/ISSS XFS Workshop is 16213, this number is represented by xfsMIBRoot within the rest of the XFS MIB documentation.

Under the xfsMIBRoot standard tree there are three main sub-trees:

xfsMIBRoot

- xfsGeneral (1)
- xfsManagedService (2)
- xfsTrap (3)
- xfsManagedApp(1000)

The xfsGeneral sub-tree contains information about the XFS environment. Under the xfsGeneral sub-tree there is a node that identifies the general version of the XFS MIB: xfsMIBV1 identifies the first version:

xfsMIBRoot(16213)

- xfsGeneral (1)
  - xfsMIBV1 (1)

The xfsManagedService sub-tree contains all information needed to define the device status and counters for each XFS class.

The xfsTrap sub-tree contains variables referenced from within the XFS Traps.

The xfsManagedApp sub-tree contains all the information relating to application management. This detail is defined in the XFS MIB application Management specification.

The XFS MIB definition is completed with a definition of how the agent must give unexpected information (i.e. a hardware error on a device) to the management centre (SNMP Manager), or in other words the definition of XFS traps.

2.1 General Information

The xfsGeneral.xfsMIBV1 sub-tree of XFS MIB tree contains essential configuration information and it allows the identification of each of the sub-trees describing the devices. It contains the following variables:

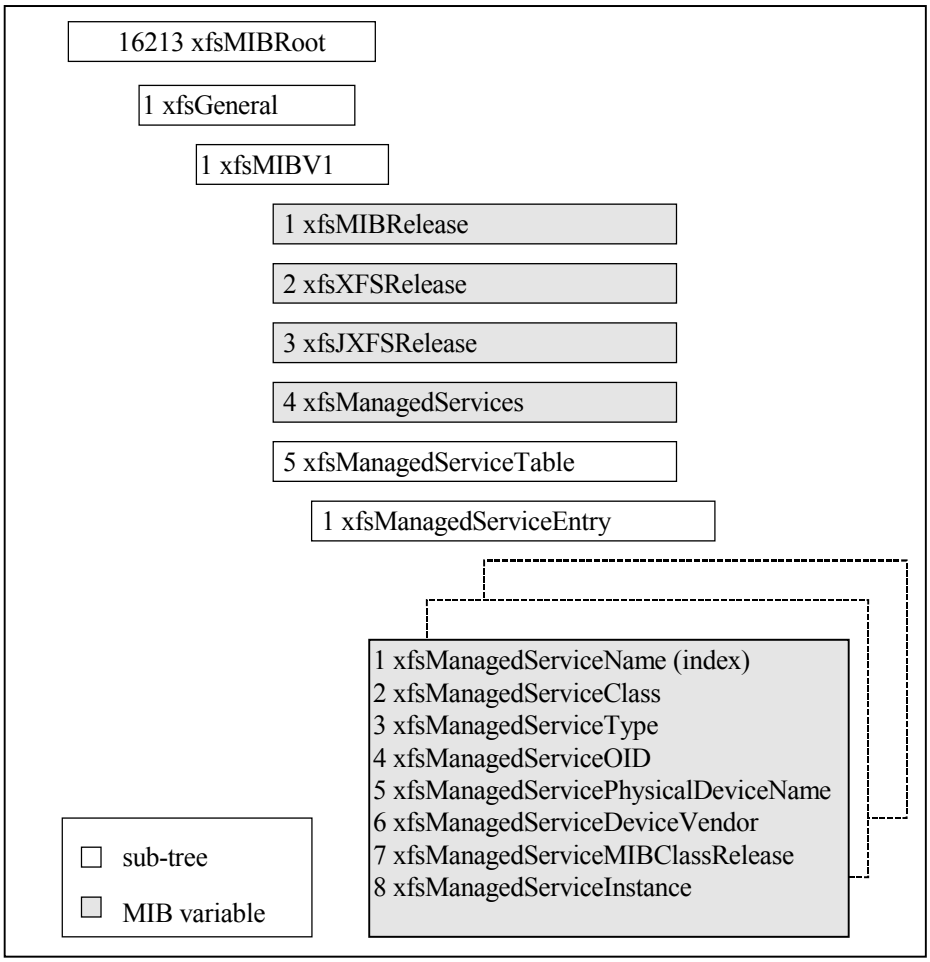
- The *xfsMIBRelease(1)* which represents the XFS MIB version. It is a 32 bit numerical field. The low-order word contains the version number, while the high-order word must be set to zero. In the low-order word, the low-order byte specifies the major version number and the high-order byte specifies the minor version number. The major version number is equal to the value of xfsMIBV1. Note: in order to allow intermediate minor revisions (e.g. between 1.10 and 1.20), the minor version number should always be expressed as two decimal digits, i.e., 1.10, 1.11, 1.20, etc.
- The *xfsXFSRelease(2)* which represents the XFS reference version, i.e. the XFS documentation release version that the MIB corresponds to. It is a 32 bit numerical field. The low-order word contains the version number, while the high-order word must be set to zero. In the low-order word, the low-order byte specifies the major version number and the high-order byte specifies the minor version number. Note: in order to allow intermediate minor revisions (e.g. between 1.10 and 1.20), the minor version number should always be expressed as two decimal digits, i.e., 1.10, 1.11, 1.20, etc.
- The *xfsJXFSRelease(3)* which represents the J/XFS MIB reference version, i.e. the J/XFS documentation release version that the MIB corresponds to. It is a 32 bit numerical field. The low-order word contains the version number, while the high-order word must be set to zero. In the low-order word, the low-order byte specifies the major version number and the high-order byte specifies the minor version number. Note: in order to allow intermediate minor revisions (e.g. between 1.10 and 1.20), the minor version number should always be expressed as two decimal digits, i.e., 1.10,

1.11, 1.20, etc. The current XFS MIB does not support J/XFS so this entry will contain 0x00000000.

- The *xfsManagedServices(4)* which represents the number of managed services installed. The managed services are enumerated from the MANAGEMENT_PROVIDERS configuration section of the XFS registry. It is a 32 bit numerical field.
- The *xfsManagedServiceTable(5)* branch defines a MIB table. Each table entry is intended to provide unchangeable information about each managed service and their associated physical devices installed on the system and contains following variables:

    - The *xfsManagedServiceName(1)*: the name of the management service. It is a Display String field.
    - The *xfsManagedServiceClass(2)*: the identifier of the XFS class. It is a 32 bit numerical field.
    - The *xfsManagedServiceType(3)*: the identifier of the XFS type. It is a 32 bit numerical field.
    - The *xfsManagedServiceOID(4)*: The OID of the sub-tree within xfsManagedService defining the management information for this class of managed service. E.g. the PTR class is represented by .1.3.6.1.4.1.16213.2. It is a Display String field.
    - The *xfsManagedServicePhysicalDeviceName(5)*: the name of the physical device or devices associated with this managed service. (See the Section "Registry configuration"). It is a Display String field. Where more than one physical device is associated with the managed service, each physical device name is comma separated.
    - The *xfsManagedServiceVendor(6)*: the vendor name of the service provider. It is a Display String field.
    - The *xfsManagedServiceMIBClassRelease(7)*: the XFS MIB class release. It is a 32 bit numerical field. The low-order word contains the version number, while the high-order word must be set to zero. In the low-order word, the low-order byte specifies the major version number and the high-order byte specifies the minor version number. The major version number is equal to the value of the XFS MIB device class release node.
    - The *xfsManagedServiceInstance(8)*: an identifier for the instance of the device class. It is a 32 bit numerical field. Its value is determined by the agent.

The following picture below shows the structure of the General branch of the XFS MIB release one. The *xfsManagedServiceTable* entry is indexed by *xfsManagedServiceName*.



As an example, the identifier of the value of *xfsManagedServiceVendor* for a card reader with an *xfsManagedServiceName equal* to "MCRW1" is as follows:

Character	M	C	R	W	1
ASCII HEX	4D	43	52	57	31
ASCII Decimal	77	67	82	87	49

NOTE: SNMP OID representation of strings consists of a length field specifying the number of characters in the string followed by the ASCII code in decimal for each character in the string. Therefore the OID of the above example is:

.xfsMIBRoot.1.1.5.1.6. 5.77.67.82.87.49

2.2 Managed Service Information

The managed service sub-tree, xfsManagedService, is located under the XFS MIB root, XFSMIBRoot. It contains one sub-tree for each XFS class.

- xfsPTR (1)
- xfsIDC (2)
- xfsCDM (3)
- xfsPIN (4)
- xfsCHK (5)
- xfsDEP (6)
- xfsTTU (7)
- xfsSIU (8)
- xfsVDM (9)
- xfsCAM (10)
- xfsALM (11)
- xfsCEU (12)
- xfsCIM (13)

The definition of every class specific sub-tree can be found in the following documents:

XFS MIB device specific definitions – PTR Device Class
XFS MIB device specific definitions – IDC Device Class
XFS MIB device specific definitions – CDM Device Class
XFS MIB device specific definitions – PIN Device Class
XFS MIB device specific definitions – CHK Device Class
XFS MIB device specific definitions – DEP Device Class
XFS MIB device specific definitions – TTU Device Class
XFS MIB device specific definitions – SIU Device Class
XFS MIB device specific definitions – VDM Device Class
XFS MIB device specific definitions – CAM Device Class
XFS MIB device specific definitions – ALM Device Class
XFS MIB device specific definitions – CEU Device Class
XFS MIB device specific definitions – CIM Device Class

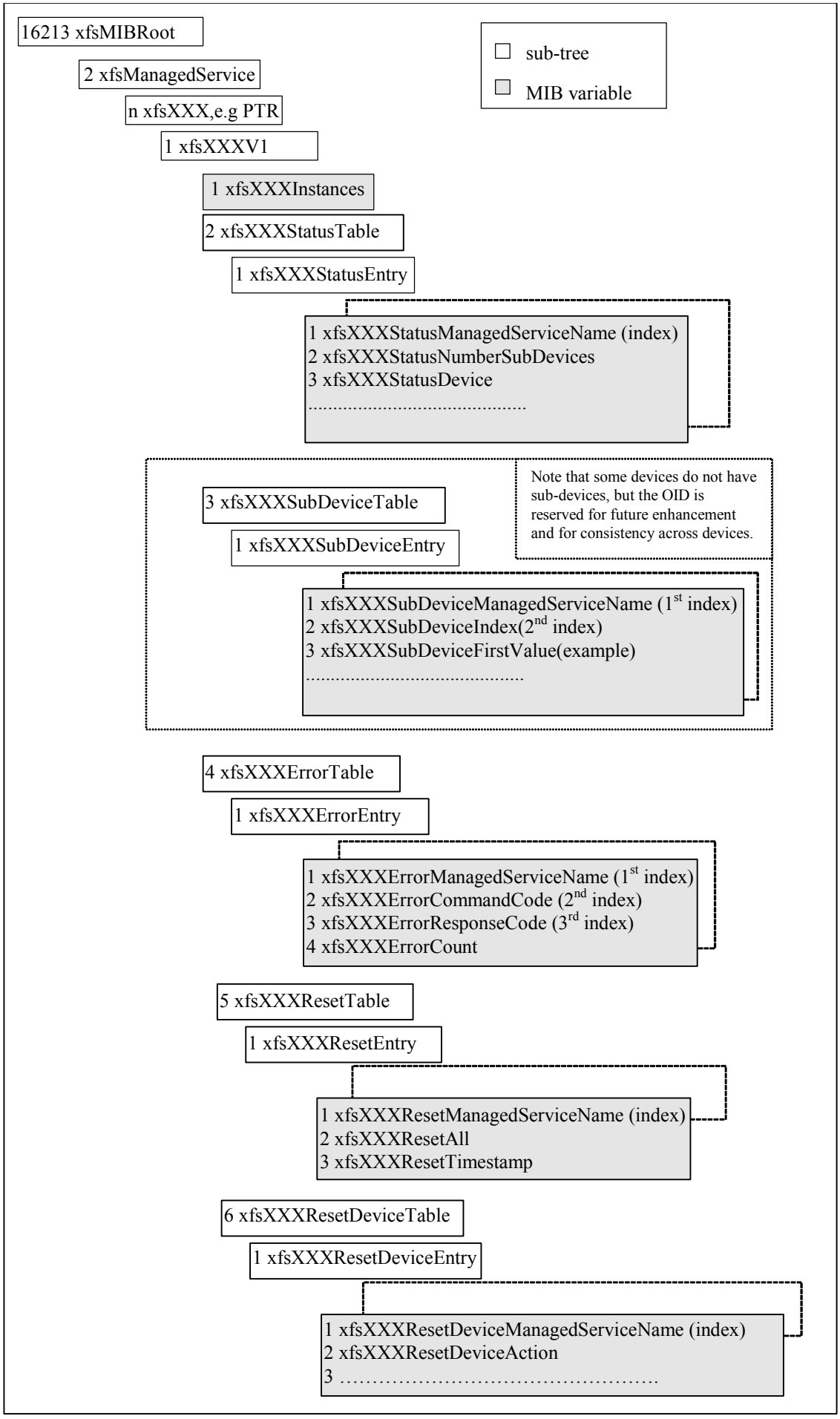
Under each class sub-tree there is a node that identifies the version of the XFS MIB device class release. For example:

xfsMIBRoot
- xfsManagedService (2)
  - xfsPTR (1)
    - xfsPTRV1 (1)

For each XFS class, the version one sub-tree has the following variables defined:

- *xfsXXXInstances(1)* number that represents the number of managed services (for the XXX class) installed on the XFS subsystem. It is a 32 bit numerical field.
- *xfsXXXStatusTable(2)* defines a set of MIB status tables, one for each managed service of the selected XXX XFS class. This table contains the status information for the managed service.
- *xfsXXXSubDeviceTable(3)* defines a set of MIB status tables for sub devices, one for each sub-device of the selected XXX XFS class. Typically this table is used to contain the status information for cash units on a CDM or CIM device.
- *xfsXXXErrorTable(4)* defines a set of MIB error tables, one for each managed service of the selected XXX XFS class. This table contains counts representing how often each of the possible response codes for each command has been generated.
- *xfsXXXResetTable(5)* defines a set of MIB reset tables, one for each managed service of the selected XXX XFS class. This table allows all of the response counters for a managed service to be reset to zero.
- *xfsXXXResetDeviceTable(6)* defines a set of MIB reset device tables, one for each managed service of the selected XXX XFS class. This table allows the device to be reset remotely.

The following picture shows the structure of the *xfsXXXV1* sub-tree of the XFS MIB as an example of an xfsManagedService (2) branch.


16213 xfsMIBRoot
2 xfsManagedService
n xfsXXX,e.g PTR
1 xfsXXXV1
1 xfsXXXInstances
2 xfsXXXStatusTable
1 xfsXXXStatusEntry
1 xfsXXXStatusManagedServiceName (index)
2 xfsXXXStatusNumberSubDevices
3 xfsXXXStatusDevice
.....
sub-tree
MIB variable
Note that some devices do not have sub-devices, but the OID is reserved for future enhancement and for consistency across devices.
3 xfsXXXSubDeviceTable
1 xfsXXXSubDeviceEntry
1 xfsXXXSubDeviceManagedServiceName (1st index)
2 xfsXXXSubDeviceIndex(2nd index)
3 xfsXXXSubDeviceFirstValue(example)
.....
4 xfsXXXErrorTable
1 xfsXXXErrorEntry
1 xfsXXXErrorManagedServiceName (1st index)
2 xfsXXXErrorCommandCode (2nd index)
3 xfsXXXErrorResponseCode (3rd index)
4 xfsXXXErrorCount
5 xfsXXXResetTable
1 xfsXXXResetEntry
1 xfsXXXResetManagedServiceName (index)
2 xfsXXXResetAll
3 xfsXXXResetTimestamp
6 xfsXXXResetDeviceTable
1 xfsXXXResetDeviceEntry
1 xfsXXXResetDeviceManagedServiceName (index)
2 xfsXXXResetDeviceAction
3 ……

2.2.1 xfsXXXStatusTable

The status table provides access to the device status values (as defined in the class specific MIB documentation) and is indexed through a single parameter, *xfsXXXStatusManagedServiceName*. This information returned in this table is obtained from the CEN XFS WFS_INF_XXX_STATUS command.

The following status table entries are common to all device classes, the other entries are defined in the class specific MIB documentation.

- The *xfsXXXStatusManagedServiceName(1)*: the name of the managed service. It is the index to the status table and is a Display String field.
- The *xfsXXXStatusNumberSubDevices(2)*: the number of sub-device for this instance. It is a 32 bit numerical field.
- The *xfsXXXStatusDevice(3)*: The device status value for the managed service.

The *xfsXXXStatusManagedServiceName* parameter corresponds to the value of *xfsMIBRoot.xfsGeneral. xfsMIBV1.xfsManagedServiceTable.xfsManagedServiceEntry.xfsManagedServiceName* in the general table. E.g. "Printer1".

All device status variables are read-only.

As an example, the identifier for the device status value of *xfsPTRStatusDevice(3)* for a device with managed service name equal to "Printer1" is as follows:

Character	P	r	i	n	t	e	r	1
ASCII Hex	50	72	69	6E	74	65	72	31
ASCII Decimal	80	114	105	110	116	101	114	49

NOTE: SNMP OID representation of strings consists of a length field specifying the number of characters in the string followed by the ASCII code in decimal for each character in the string. Therefore the OID of the above example is:

xfsMIBRoot.2.1.1.2.1.3.8.80.114.105.110.116.101.114.49

2.2.2 xfsXXXSubDeviceTable

The sub-device table provides access to the device status values for sub devices within a service class. For example, on a CDM the cash units are represented as sub-devices. However, not all service classes require sub-devices, in this case the *xfsXXXStatusTable* entry, *xfsStatusNumberSubDevices* will be zero and although the sub-device table will exist it will have no entries.

The following sub-device status table entries are common to all sub-devices, the other entries are defined in the class specific MIB documentation.

- The *xfsXXXSubDeviceManagedServiceName(1)*: the name of the managed service that the sub-device belongs to. It is the index to the sub-device status table and is a Display String field.
- The *xfsXXXSubDeviceIndex(2)*: the 2nd index identifying the specific sub-device within the managed service. It is a 32 bit numerical field, with a valid range from 1 to the number of sub devices defined within *xfsXXXStatusNumberSubDevices* variable within the corresponding managed service status table entry.

The *xfsXXXStatusManagedServiceName* parameter corresponds to the value of x*fsMIBRoot.xfsGeneral.* xfsMIBV1*.xfsManagedServiceTable.xfsManagedServiceEntry.xfsManagedServiceName* in the general table. E.g. "CDM1".

All sub-device status variables are read-only.

As an example, the identifier for the Type, xfsCDMSubDeviceType(3), value within the first Cash Unit (i.e. the first sub-device) for a device with managed service name equal to “CDM1” is as follows:

Character	C	D	M	1
ASCII Hex	43	44	4D	31
ASCII Decimal	67	68	77	49

NOTE: SNMP OID representation of strings consists of a length field specifying the number of characters in the string followed by the ASCII code in decimal for each character in the string. Therefore the OID of the above example is:

xfsMIBRoot.2.3.1.3.1.3.4.67.68.77.49.1

2.2.3 xfsXXXErrorTable

The *xfsXXXErrorTable* provides access to command response counters supported by a device class. The error table contains the set of counters for every combination of executable command and associated response that the service provider supports for the managed service. Selection of the required counter is made by specifying the managed service name, command code and response code through the following parameters

xfsXXXErrorManagedServiceName
xfsXXXErrorCommandCode
xfsXXXErrorResponseCode

The *xfsXXXErrorTable* is defined as:

- *xfsXXXErrorManagedServiceName(1*) which provides the primary index to the service in question. It is a Display String field. The *xfsXXXErrorManagedServiceName* parameter corresponds to the value of *xfsMIBRoot.xfsGeneral.* xfsMIBV1*.xfsManagedServiceTable.xfsManagedServiceTableEntry.xfsManagedServiceName* in the general table. E.g. “Printer1”.
- *xfsXXXErrorCommandCode(2)* is an index which identifies the command code that the response code relates to, e.g. WFS_CMD_PTR_CONTROL_MEDIA (301). It is a 32 bit numerical field.
- *xfsXXXErrorResponseCode(3*) is an index which identifies the response code that the count is required for. It is the absolute value of the error code e.g. WFS_ERR_PTR_NOMEDIAPRESENT (-302) is represented by 302. It is a 32 bit numerical field
- *xfsXXXErrorCount*(4)is the count of the number of times that a particular response code has been generated while executing a specific command. It is a 32 bit numerical field.

The *xfsXXXErrorCount(4)* variable is read-write, the other table entries are indices. Issue of a Set command on a specific counter with value x will result in the individual counter being set to value x.

As an example, the identifier for the error count value for the WFS_ERR_PTR_NOMEDIAPRESENT(-302) error returned from the WFS_CMD_PTR_CONTROL_MEDIA(301) command for a device with managed service name equal to “Printer1” is as follows:

xfsMIBRoot.2.1.1.4.1.4.8.80.114.105.110.116.101.114.49.301.302

2.2.4 xfsXXXResetTable

The *xfsXXXResetTable* provides the means of resetting all counters within the error table and is indexed by a single variable:

xfsXXXResetManagedServiceName

The *xfsXXXResetTable* is defined as:

- *xfsXXXResetManagedServiceName(1)* which provides the index to the service in question. It is a Display String field. The *xfsXXXResetManagedServiceName* parameter corresponds to the value of *xfsMIBRoot.xfsGeneral.* xfsMIBV1*.xfsManagedServiceTable.xfsManagedServiceTableEntry.xfsManagedServiceName* in the general table. E.g. “Printer1”.
- *xfsXXXResetAll(2)* is a read-write 32 bit numerical variable. Issue of a Set command on the *xfsXXXResetAll* variable with value 0 (zero) will result in all counters for the managed service being reset to value 0 (zero). Any other value will be ignored and the counters will remain unchanged A query of the *xfsIDCResetAll* variable will return 0 (zero).
- *xfsXXXResetTimestamp(3)* is a read-only variable which represents the UTC and bias for local translation of the date and time when the counters in the error table was reset. It is a Display String field. The data is formatted in the following way: "DD/MM/YYYY HH:MM:SS +ZZZ” where DD/MM/YYYY HH:MM:SS is the local date and time. ZZZ is the bias, which is the difference, in minutes, between Co-ordinated Universal Time (UTC) and local time

As an example, all the error counts can be reset for a device with managed service name equal to “Printer1” by setting the value zero in the *xfsPTRResetAll(2)* variable represented by:

xfsMIBRoot.2.1.1.5.1.2.8.80.114.105.110.116.101.114.49

2.2.5 XFS XXX Reset Device Table

The *xfsXXXResetDeviceTable*(6) is indexed by the single variable, *xfsXXXResetDeviceManagedServiceName*. This table contains variables which monitor and control the execution of the reset request.

The *xfsXXXResetDeviceAction* variable is used to initiate a reset. Setting this variable will cause the following to happen

1. The SNMP agent will determine if a Device Reset is allowed by checking the *RemoteDeviceResetAllowed* configuration flag (see section 3.1). If it is not allowed then the flow continues with step 5, otherwise the flow continues with step 2.
2. Exclusive access to the device will be obtained
3. A WFS_CMD_XXX_RESET command will be issued
4. Exclusive access to the device will be relinquished when the WFS_CMD_XXX_RESET command completes. Note: Exclusive access must be relinquished as soon as possible and implemented in such a way that deadlocks are avoided.
5. A *xfsXXXResetDeviceCompleteTrap* trap will be generated to report the result of the Device Reset request

The *xfsXXXResetDeviceMediaControl* variable is used to report how any media found within the device is handled.

The *xfsXXXResetDeviceTable*(6) is defined as:

- *xfsXXXResetDeviceManagedServiceName(1)* which provides the index to the service in question. It is a Display String field. The *xfsXXXResetDeviceManagedServiceName* parameter corresponds to the value of *xfsMIBRoot.xfsGeneral.xfsMIBV1.xfsManagedServiceTable.xfsManagedServiceEntry.xfsManagedServiceName* in the general table. E.g. “Printer1”.

- *xfsXXXResetDeviceAction(2))* is a read-write variable. Issue of a Set command on the *xfsXXXResetDeviceAction* variable with value executeReset(1) will result in the device being reset as described above.
- *xfsXXXResetDeviceMediaControl(3))* is a read only variable. This variable reports how any media found within the device is handled. The value of the *xfsXXXResetDeviceMediaControl* variable is configured through the *ResetDeviceMediaControl* configuration setting (see section 3.2). If this value is not configured then the variable defaults to the value that indicates that the Service Provider is responsible for media control. The detailed device specific media control information (e.g. PTR bin to retract media to) is configured through local SNMP Agent configuration.
- *xfsXXXResetDeviceStatus(4))* is a read only variable This variable can be used to check if a reset operation is still in progress. It is set when the reset is initiated and cleared when the reset command completes.

As an example, the device with managed service name equal to "Printer1" is reset by setting the *xfsXXXResetDeviceAction* variable represented by:

xfsMIBRoot.2.1.1.6.1.2. 8.80.114.105.110.116.101.114.49

2.3 XFS Traps

In order to generate traps, the SNMP agent should register itself for receiving all error condition notifications from all managed services installed on a system.
Device status changes are reported by the Device Status Change system event and error conditions are reported by the Hardware and Software Error system events. When the agent receives one of the above events, a trap is generated. All data associated with the trap is retrieved by the SNMP agent using the managed service name identified from the above events, while both the related *hService* and *RequestID* parameters are not meaningful.

The SNMP XFS agent must use the XFS functions for memory management. In particular, the SNMP XFS Agent calls the *WFSFreeResult* function to free the data related to the events described in this chapter. The XFS functions for memory management are described in the XFS Programmer's Reference.

See the Registry configuration section for a detailed description of how the managed services are configured and how the physical device name is associated with the management data.

The XFS traps are identified with the generic parameter value 6 (standard SNMP) and the specific XFS parameter value that defines the specific trap managed.

The following specific trap values are defined for XFS traps that are common across all device classes. The detail for these traps are defined in this document.

- xfsDSCTrap(1)
- xfsErrorTrap (2)
- xfsThreshold(3)

The following specific trap values are defined for XFS traps that are specific to each device class. These traps are defined in the class specific MIB documentation. There are two classes of device specific Traps, Detailed Device Status Change Traps and Sub Device Status Change Traps. Specific trap values 101 to 200 are reserved for Detailed Device Status Change Traps and Specific trap values 201 to 300 are reserved for Sub Device Status Change Traps.

The following device specific Detailed Device Status Change traps are generated in addition to the generic Device Status change trap above on version 1.1 of the MIB and above.

- xfsPTRDetailedDSC (101)
- xfsIDCDetailedDSC (102)
- xfsCDMDetailedDSC (103)
- xfsPINDetailedDSC (104)
- xfsCHKDetailedDSC (105)
- xfsDEPDetailedDSC (106)

- xfsTTUDetailedDSC (107)
- xfsSIUDetailedDSC (108)
- xfsVDMDetailedDSC (109)
- xfsCAMDetailedDSC (110)
- xfsALMDetailedDSC (111)
- xfsCEUDetailedDSC (112)
- xfsCIMDetailedDSC (113)

The following Sub Device Status Change Traps are generated on version 1.1 of the MIB and above.

- xfsPTRSubDeviceSC (201)
- xfsIDCSubDeviceSC (202)
- xfsCDMSubDeviceSC (203)
- xfsPINSubDeviceSC (204)
- xfsCHKSubDeviceSC (205)
- xfsDEPSubDeviceSC (206)
- xfsTTUSubDeviceSC (207)
- xfsSIUSubDeviceSC (208)
- xfsVDMSubDeviceSC (209)
- xfsCAMSubDeviceSC (210)
- xfsALMSubDeviceSC (211)
- xfsCEUSubDeviceSC (212)
- xfsCIMSubDeviceSC (213)

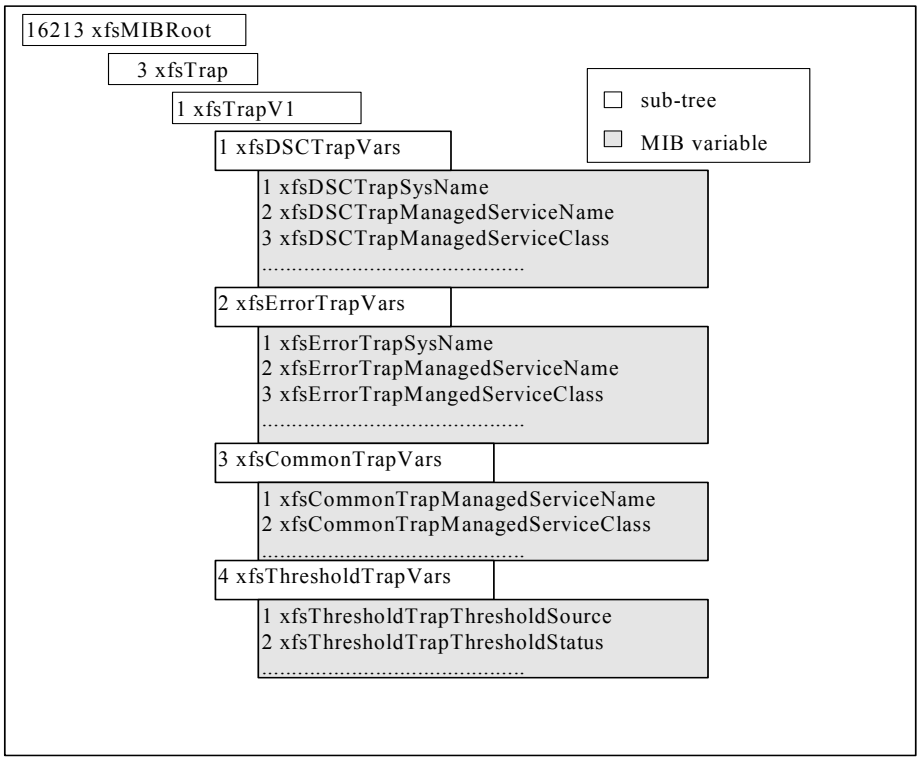
Note only the PTR, CDM and CIM currently have sub-devices, the other traps are reserved for future use.

The following Reset Device Traps are generated on version 1.1 of the MIB and above.

- xfsPTRResetDeviceCompleteTrap (301)
- xfsIDCResetDeviceCompleteTrap (302)
- xfsCDMResetDeviceCompleteTrap (303)
- xfsPINResetDeviceCompleteTrap (304)
- xfsCHKResetDeviceCompleteTrap (305)
- xfsDEPResetDeviceCompleteTrap (306)
- xfsTTUResetDeviceCompleteTrap (307)
- xfsSIUResetDeviceCompleteTrap (308)
- xfsVDMResetDeviceCompleteTrap (309)
- xfsCAMResetDeviceCompleteTrap (310)
- xfsALMResetDeviceCompleteTrap (311)
- xfsCEUResetDeviceCompleteTrap (312)
- xfsCIMResetDeviceCompleteTrap (313)

The Application Management MIB uses the specific trap value of 1000 to report changes in the states of the applications.

The traps generated within the XFS MIB contain variable bindings which make reference to variables defined within the XFS MIB. The following picture shows the structure of the sub-tree referenced by the XFS traps.



Inside each trap there is a variable binding list, defined in the following chapters and in the device specific MIB documentation. The variable binding list contains all the information associated with an alarm to be sent to the remote server (SNMP Manager). The XFS Agent sets the variables with the values of the corresponding fields delivered by the system events.

In the above MIB tree, the xfsCommonTrapVars sub-tree contains variables that are common to all new events added to the MIB specification since version 1.0. The variable binding list within new events reference these variables for common elements and then reference trap specific variables as required by the trap in question.

2.3.1 Device Status Changes

Status changes of managed services are reported as system events to the XFS Agent. In case of XFS, the definition of the event data structure can be found in the XFS 3.0 API/SPI Programmer's Reference document.

It is only the top level status change within a managed service that is reported through a Device Status Change Event, i.e. the trap is generated when the fwDevice value in the WFS_INF_XXX_STATUS response has changed.

The Device Status Change Trap contains a specific variable binding for each element in the trap. The following section describes the trap variables and trap format.

2.3.1.1 XFS Device Status Change Trap Format

All variables referenced within the Device Status Change Trap are contained within the sub-tree defined by the following elements.

xfsMIBRoot	16213
xfsTrap	3
xfsTrapV1	1
xfsDSCTrapVars	1

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapSysName (1)
This variable binding contains the system generating the alarm, it is a Display String field. It corresponds to lpszWorkstationName in the device status change event data from the service provider.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapManagedServiceName (2)
This variable binding represents the managed service name generating the alarm, it is a Display String field. The agent derives this field from the device status change event.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapManagedServiceClass (3)
This variable binding represents the XFS service class identifier generating the alarm, it is a 32-bit integer (INT32). It corresponds to the class identifier for the class name. The class name is identified from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<Managed*ServiceName*>\class. This ID matches the class OID branch number i.e. PTR=1, IDC=2, CDM=3, etc.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapManagedServiceClassName (4)
This variable binding represents the XFS service class name generating the alarm, it is a Display String field. It corresponds to the three character representation of the XFS device class name, and it is useful for human interpretation of a trap. The class name is identified from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\class.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapManagedServiceType (5)
This variable binding represents the XFS type identifier generating the alarm, it is a 32-bit integer (INT32). It corresponds to the type identifier as defined in the WFS_INF_XXX_CAPABILITIES.fwType field, or zero if device class does not support this field.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapManagedServiceOid (6)
This variable binding represents the OID of the sub-tree within xfsManagedService defining the management information for this class of managed service. This variable, along with the managed service name as an index, prevents the need for additional querying to find the service specific MIB branch. The value is provided by the SNMP agent. E.g. the PTR MIB class is represented by .1.3.6.1.4.1.16213.2.1

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapPhysicalDeviceName (7)
This variable binding represents the physical device name or names associated with the managed service generating the alarm, it is a Display String field. It corresponds to the physical device name or names identified by the managed service. The managed service name is used to identify the physical device name or names, from registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\PhysicalDeviceName. Multiple physical device names are comma separated. E.g. "ABC Printer Engine, ABC Transport".

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapDeviceVendor (8)
This variable binding represents the XFS device vendor name of the device generating the alarm, it is a Display String field. It corresponds to the vendor name for the service provider. The service provider is identified from the managed service name and the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\ ServiceProvider.
The service provider name is then used to identify the vendor, from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\SERVICE_PROVIDERS\<*ServiceProviderName*>\vendor_name.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapMIBVersion (9)
This variable binding represents the XFS MIB version of the device generating the alarm, it is a Display String field. It corresponds to the XFS MIB version for the managed service. The managed service name is used to identify the XFS MIB version, from registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\ MibVersion.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapEvent (10)
In case of XFS this variable binding represents the XFS event generating the alarm, it is a 32-bit integer (INT32). It corresponds to u.dwEventID in the event data from the service provider.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapDate (11)
This variable represents the UTC and bias for local translation of the date and time when the event was generated. It is a Display String field. The data is formatted in the following way: "DD/MM/YYYY HH:MM:SS +ZZZ" where DD/MM/YYYY HH:MM:SS is the local date and time. ZZZ is the bias, which is the difference, in minutes, between Co-ordinated Universal Time (UTC) and local time.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapSPVersion (12)
This variable represents the vendor-defined version of the service provider generating the alarm, it is a Display String field. The service provider is identified from the managed service name and the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\ ServiceProvider.
The service provider name is then used to identify the version, from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\SERVICE_PROVIDERS\<*ServiceProviderName*>\version.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapManagedServiceStatus (13)
This variable binding represents the current state of the physical device managed by the service, and corresponds to dwState in the event data from the service provider. It is a 32 bit integer (INT32).

2.3.1.2 XFS Device Status Change Trap: an example

As an example, the following variable binding list represents a device status change trap (6, 1) generated from a generic XFS SST system to send an alarm to the SNMP Manager for paper finished on the physical device PTR of type receipt and managed service name “Printer1”.

xfsMIBRoot.3.1.1.1	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapSysName)
	“SST System 1”
xfsMIBRoot.3.1.1.2	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapManagedServiceName)
	“Printer1”
xfsMIBRoot.3.1.1.3	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapManagedServiceClass)
	1 (WFS_SERVICE_CLASS_PTR)
xfsMIBRoot.3.1.1.4	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapManagedServiceClassName)
	“PTR”
xfsMIBRoot.3.1.1.5	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapManagedServiceType)
	1 (WFS_PTR_TYPERECEIPT)
xfsMIBRoot.3.1.1.6	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapManagedServiceOid)
	“.1.3.6.1.4.1.16213.2.1”
xfsMIBRoot.3.1.1.7	(xfsMIBRoot.xfsTrap. xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapPhysicalDeviceName)

	“ABC Corp Receipt Printer”
xfsMIBRoot.3.1.1.8	(xfsMIBRoot.xfsTrap. xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapDeviceVendor)
	“Best Printers Incorporated”
xfsMIBRoot.3.1.1.9	(xfsMIBRoot.xfsTrap. xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapMIBVersion)
	“1.00”
xfsMIBRoot.3.1.1.10	(xfsMIBRoot.xfsTrap. xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapEvent)
	4 (WFS_SYSE_DEVICE_STATUS)
xfsMIBRoot.3.1.1.11	(xfsMIBRoot.xfsTrap. xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapDate)
	“20/03/2003 15:40:53 -300”
xfsMIBRoot.3.1.1.12	(xfsMIBRoot.xfsTrap. xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapSPVersion)
	“1.23”
xfsMIBRoot.3.1.1.13	(xfsMIBRoot.xfsTrap. xfsTrapV1.xfsDSCTrapVars.xfsDSCTrapManagedServiceStatus)
	2 (WFS_STAT_DEVOFFLINE)

2.3.2 Hardware and Software Errors

Hardware and software errors are also reported as system events to the XFS Agent. In case of XFS, the definition of the event data structure can be found in the XFS 3.0 API/SPI Programmer’s Reference document.

The Error Trap contains a specific variable binding for each element in the trap. The following section describes the trap variables and trap format.

2.3.2.1 XFS Error Trap Format

All variables referenced within the Error Trap are contained within the sub-tree defined by the following elements.

xfsMIBRoot	16213
xfsTrap	3
xfsTrapV1	1
xfsErrorTrapVars	2

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapSysName (1)
The first variable binding contains the system generating the alarm, it is a Display String field. It corresponds to lpszWorkstationName in the event data from the service provider.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapManagedServiceName (2)
This variable binding represents the managed service name generating the alarm, it is Display String field. The agent derives this field from the error event.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapManagedServiceClass (3)
This variable binding represents the XFS service class identifier generating the alarm, it is a 32-bit integer (INT32). It corresponds to the class identifier for the class name. The class name is identified from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\class. This ID matches the class OID branch number i.e. PTR=1, IDC=2, CDM=3, etc.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapManagedServiceClassName (4)
This variable binding represents the XFS service class name generating the alarm, it is a Display String field. It corresponds to the three character representation of the XFS service class name, and it is useful for human interpretation of a trap. The class name is identified from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\class.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapManagedServiceType (5)

This variable binding represents the XFS type identifier generating the alarm, it is a 32-bit integer (INT32). It corresponds to the type identifier as defined in the WFS_INF_XXX_CAPABILITIES.fwType field, or zero if device class does not support this field.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapManagedServiceOid (6)
This variable binding represents the OID of the sub-tree within xfsManagedService defining the management information for this class of managed service. This variable, along with the managed service name as an index, prevents the need for additional querying to find the service specific MIB branch. The value is provided by the SNMP agent. E.g. the PTR MIB class is represented by.1.3.6.1.4.1.16213.2.1

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapPhysicalDeviceName (7)
This variable binding represents the physical device name or names associated with the managed service generating the alarm, it is a Display String field. It corresponds to the physical device name or names identified by the managed service. The managed service name is used to identify the physical device name or names, from registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\PhysicalDeviceName. Multiple physical device names are comma separated. E.g. “ABC Printer Engine, ABC Transport”

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapDeviceVendor (8)
This variable binding represents the XFS device vendor name of the device generating the alarm, it is a Display String field. It corresponds to the vendor name for the service provider. The service provider is identified from the managed service and the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\ServiceProvider.
The service provider name is then used to identify the vendor, from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\SERVICE_PROVIDERS\<*ServiceProviderName*>\vendor_name.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapMIBVersion (9)
This variable binding represents the XFS MIB version of the device generating the alarm, it is a Display String field. It corresponds to the XFS MIB version for the managed service. The managed service name is used to identify the XFS MIB version, from registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\MibVersion.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapEvent (10)
In case of XFS, this variable binding represents the XFS event generating the alarm, it is a 32-bit integer (INT32). It corresponds to u.dwEventID in the event data from the service provider..

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapDate (11)
This variable represents the UTC and bias for local translation of the date and time when the event was generated, it is a Display String field. The data is formatted in the following way: "DD/MM/YYYY HH:MM:SS +ZZZ" where DD/MM/YYYY HH:MM:SS is the local date and time. ZZZ is the bias, which is the difference, in minutes, between Co-ordinated Universal Time (UTC) and local time.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapSPVersion (12)
This variable represents the vendor-defined version of the service provider generating the alarm, it is a Display String field. The service provider is identified from the managed service name and the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\ServiceProvider.
The service provider name is then used to identify the version, from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\SERVICE_PROVIDERS\<*ServiceProviderName*>\version.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapSuggestedAction (13)
This variable binding represents the suggested action, and corresponds to dwAction in the event data from the service provider. It is a 32 bit integer (INT32).

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapDescrString (14)
This variable binding represents the description associated to the alarm (this description is vendor dependent), it is a 255 chars length string (OCTET STRING(255)). In case of XFS, it corresponds to lpbDescription in the event data from the service provider.

xfsMIBRoot. xfsTrap. xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapAppId (15)
This variable binding represents the application ID associated with the session that generated the error event, and corresponds to lpszAppID in the event data from the service provider. It is a Display String field.

2.3.2.2 XFS Error Trap: an example

As an example, the following variable binding list represents an error trap (6, 2) generated from a generic XFS SST system to send an alarm to the SNMP Manager indicating that the device needs to have the hardware cleared, a PTR of type receipt and managed service name “Printer1”.

xfsMIBRoot.3.1.2.1	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapSysName)
	“SST System 1”
xfsMIBRoot.3.1.2.2	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapManagedServiceName)
	“Printer1”
xfsMIBRoot.3.1.2.3	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapManagedServiceClass)
	1 (WFS_SERVICE_CLASS_PTR)
xfsMIBRoot.3.1.2.4	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapManagedServiceClassName)
	“PTR”
xfsMIBRoot.3.1.2.5	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapManagedServiceType)
	1 (WFS_PTR_TYPERECEIPT)
xfsMIBRoot.3.1.2.6	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapManagedServiceOid)
	“.1.3.6.1.4.1.16213.2.1”
xfsMIBRoot.3.1.2.7	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapPhysicalDeviceName)
	“ReceiptPrinter1”
xfsMIBRoot.3.1.2.8	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapDeviceVendor)
	“Best Printers Incorporated”
xfsMIBRoot.3.1.2.9	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapMIBVersion)
	“1.00”
xfsMIBRoot.3.1.2.10	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapEvent)
	2 (WFS_SYSE_HARDWARE_ERROR)
xfsMIBRoot.3.1.2.11	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapDate)
	“20/03/2003 15:40:53 -300”
xfsMIBRoot.3.1.2.12	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapSPVersion)
	“1.23”
xfsMIBRoot.3.1.2.13	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapSuggestedAction)
	0x0008 (WFS_ERR_ACT_HWCLEAR)
xfsMIBRoot.3.1.2.14	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapDescrString)
	“Clear Transport”
xfsMIBRoot.3.1.2.15	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsErrorTrapVars.xfsErrorTrapAppId)
	“Consumer Application”

2.3.3 Common Trap Variables

All new traps added to the XFS MIB after version 1.0 make reference to the following variables for information that is common to traps. This prevents the need for multiple definitions for variables that appear in many traps across many device classes.

The variables common to many traps are contained within the sub-tree defined by the following elements.

xfsMIBRoot	16213
xfsTrap	3
xfsTrapV1	1

xfsCommonTrap 3

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapSysName (1)
This variable binding contains the system generating the alarm, it is a Display String field. It corresponds to lpszWorkstationName in device status change and error events from the service provider.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceName (2)
This variable binding represents the XFS managed service name generating the alarm, it is a Display String field. The agent derives this field from the XFS event causing the trap.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceClass (3)
This variable binding represents the XFS service class identifier generating the alarm, it is a 32-bit integer (INT32). It corresponds to the class identifier for the class name. The class name is identified from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<Managed*ServiceName*>\class. This ID matches the class OID branch number i.e. PTR=1, IDC=2, CDM=3, etc.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceClassName(4)
This variable binding represents the XFS service class name generating the alarm, it is a Display String field. It corresponds to the three character representation of the XFS device class name, and it is useful for human interpretation of a trap. The class name is identified from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\class.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceType (5)
This variable binding represents the XFS type identifier generating the alarm, it is a 32-bit integer (INT32). It corresponds to the type identifier as defined in the WFS_INF_XXX_CAPABILITIES.fwType field, or zero if device class does not support this field.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceOid (6)
This variable binding represents the OID of the sub-tree within xfsManagedService defining the management information for this class of managed service. This variable, along with the managed service name as an index, prevents the need for additional querying to find the service specific MIB branch. The value is provided by the SNMP agent. E.g. the PTR MIB class is represented by .1.3.6.1.4.1.16213.2.1

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapPhysicalDeviceName (7)
This variable binding represents the physical device name or names associated with the managed service generating the alarm, it is a Display String field. It corresponds to the physical device name or names identified by the managed service. The managed service name is used to identify the physical device name or names, from registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\PhysicalDeviceName. Multiple physical device names are comma separated. E.g. “ABC Printer Engine, ABC Transport”.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapDeviceVendor (8)
This variable binding represents the XFS device vendor name of the device generating the alarm, it is a Display String field. It corresponds to the vendor name for the service provider. The service provider is identified from the managed service name and the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\ServiceProvider.
The service provider name is then used to identify the vendor, from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\SERVICE_PROVIDERS\<*ServiceProviderName*>\vendor_name.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapMIBVersion (9)
This variable binding represents the XFS MIB version of the device generating the alarm, it is a Display String field. It corresponds to the XFS MIB version for the managed service. The managed service name is used to identify the XFS MIB version, from registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\MibVersion.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapEvent (10)
In case of XFS this variable binding represents the XFS threshold event generating the alarm, it is a 32-bit integer (INT32). It corresponds to the message identifier associated with the XFS event causing the trap.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapDate (11)

This variable represents the UTC and bias for local translation of the date and time when the event was generated. It is a Display String field. The data is formatted in the following way: "DD/MM/YYYY HH:MM:SS +ZZZ" where DD/MM/YYYY HH:MM:SS is the local date and time. ZZZ is the bias, which is the difference, in minutes, between Co-ordinated Universal Time (UTC) and local time.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapSPVersion (12)

This variable represents the vendor-defined version of the service provider generating the alarm, it is a Display String field. The service provider is identified from the managed service name and the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\ ServiceProvider.

The service provider name is then used to identify the version, from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\SERVICE_PROVIDERS\<*ServiceProviderName*>\version.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapResetRequestResult (13)

This variable binding contains a value indicating if the device reset was executed, and if not provides a reason. It does not report the status of the device. The possible values are:

zero	the reset was executed successfully (however the device may not be operational, see the device status fields).
one	the reset was rejected because exclusive access could not be obtained
two	the reset was rejected because Device Resets are disabled on this terminal (see Section 3.1)
negative	the reset request was executed but failed and the value corresponds to the XFS error code

2.3.4 Threshold Status Changes

This trap is generated when a device threshold event occurs (WFS_USRE_XXX_THRESHOLD). The trap uniquely identifies the underlying threshold event via the following fields

xfsEvent – describes the underlying XFS threshold event that has occurred.

xfsThresholdSource – Describes the source component of the device (as described in the associated Threshold event) that this change of state is associated with when a threshold can be generated from multiple sources within a device. For example, the PTR supports threshold values with multiple sources, i.e. a threshold for one of a number of retract bin numbers or one of a number of paper sources. In this case, this field will contain a value that represents the specific source of the event. It corresponds to the source field within the threshold event identified by the xfsEvent above. A value of zero indicates that the event data did not include this detail.

On the CDM and CIM device classes this field corresponds to the sub-device index for the cash unit that has generated the event.

xfsThresholdStatus – The new value of the threshold as described in the Threshold event

2.3.4.1 XFS Threshold Specific Variables

Variables that are specific to the Threshold trap are contained within the sub-tree defined by the following elements.

xfsMIBRoot	16213
xfsTrap	3
xfsTrapV1	1
xfsThresholdTrap	4

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsThresholdTrapVars.xfsThresholdTrapThresholdSource (1)

This variable binding represents the specific source of the threshold event when a specific threshold can be generated from multiple sources within a device. For example, the PTR supports threshold values with multiple sources, i.e. a threshold for one of a number of retract bin numbers or one of a number of paper sources. In this case, this field will contain a value that represents the specific source of the event. It corresponds to the source field within the threshold event identified by the xfsEvent above.

On the CDM and CIM device classes this field corresponds to the sub-device index for the cash unit that has generated the event.

However, most device thresholds have a single source within a device, and in this case this value will always report 0. It is a 32 bit integer (INT32).

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsThresholdTrapVars.xfsThresholdTrapThresholdStatus (2)
This variable binding represents the current threshold state within the device managed by the service, and corresponds to threshold value in the event data from the service provider. It is a 32 bit integer (INT32).

2.3.4.2 XFS Threshold Trap Format

The following are the variable bindings contained within the XFS Threshold Trap.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceName (1)
This variable binding represents the XFS managed service name generating the alarm, it is a Display String field. The agent derives this field from the threshold event.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceClass (2)
This variable binding represents the XFS service class identifier generating the alarm, it is a 32-bit integer (INT32). It corresponds to the class identifier for the class name. The class name is identified from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<Managed*ServiceName*>\class. This ID matches the class OID branch number i.e. PTR=1, IDC=2, CDM=3, etc.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceClassName(3)
This variable binding represents the XFS service class name generating the alarm, it is a Display String field. It corresponds to the three character representation of the XFS device class name, and it is useful for human interpretation of a trap. The class name is identified from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\class.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceType (4)
This variable binding represents the XFS type identifier generating the alarm, it is a 32-bit integer (INT32). It corresponds to the type identifier as defined in the WFS_INF_XXX_CAPABILITIES.fwType field, or zero if device class does not support this field.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceOid (5)
This variable binding represents the OID of the sub-tree within xfsManagedService defining the management information for this class of managed service. This variable, along with the managed service name as an index, prevents the need for additional querying to find the service specific MIB branch. The value is provided by the SNMP agent. E.g. the PTR MIB class is represented by .1.3.6.1.4.1.16213.2.1

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapPhysicalDeviceName (6)
This variable binding represents the physical device name or names associated with the managed service generating the alarm, it is a Display String field. It corresponds to the physical device name or names identified by the managed service. The managed service name is used to identify the physical device name or names, from registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\PhysicalDeviceName. Multiple physical device names are comma separated. E.g. “ABC Printer Engine, ABC Transport”.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapDeviceVendor (7)
This variable binding represents the XFS device vendor name of the device generating the alarm, it is a Display String field. It corresponds to the vendor name for the service provider. The service provider is identified from the managed service name and the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\ServiceProvider.
The service provider name is then used to identify the vendor, from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\SERVICE_PROVIDERS\<*ServiceProviderName*>\vendor_name.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapMIBVersion (8)
This variable binding represents the XFS MIB version of the device generating the alarm, it is a Display String field. It corresponds to the XFS MIB version for the managed service. The managed service name is used to identify the XFS MIB version, from registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\MibVersion.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapEvent (9)

In case of XFS this variable binding represents the XFS threshold event generating the alarm, it is a 32-bit integer (INT32). It corresponds to the message identifier associated with the device User event generated by the service provider for this threshold change.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapDate (10)
This variable represents the UTC and bias for local translation of the date and time when the event was generated. It is a Display String field. The data is formatted in the following way: "DD/MM/YYYY HH:MM:SS +ZZZ" where DD/MM/YYYY HH:MM:SS is the local date and time. ZZZ is the bias, which is the difference, in minutes, between Co-ordinated Universal Time (UTC) and local time.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapSPVersion (11)
This variable represents the vendor-defined version of the service provider generating the alarm, it is a Display String field. The service provider is identified from the managed service name and the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS\<*ManagedServiceName*>\ ServiceProvider.
The service provider name is then used to identify the version, from the registry value HKEY_LOCAL_MACHINE\SOFTWARE\XFS\SERVICE_PROVIDERS\<*ServiceProviderName*>\version.

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsThresholdTrapVars.xfsThresholdTrapThresholdSource (12)
This variable binding represents the specific source of the threshold event when a specific threshold can be generated from multiple sources within a device. For example, the PTR supports threshold values with multiple sources, i.e. a threshold for one of a number of retract bin numbers or one of a number of paper sources. In this case, this field will contain a value that represents the specific source of the event. It corresponds to the source field within the threshold event identified by the xfsEvent above.

On the CDM and CIM device classes this field corresponds to the sub-device index for the cash unit that has generated the event.

However, most device thresholds have a single source within a device, and in this case this value will always report 0. It is a 32 bit integer (INT32).

xfsMIBRoot.xfsTrap.xfsTrapV1.xfsThresholdTrapVars.xfsThresholdTrapThresholdStatus (13)
This variable binding represents the current threshold state within the device managed by the service, and corresponds to threshold value in the event data from the service provider. It is a 32 bit integer (INT32).

2.3.4.3 XFS Threshold Trap: an example

As an example, the following variable binding list represents a threshold trap (6,3) generated from a generic XFS SST system to send an alarm to the SNMP Manager indicating that the device needs to have the hardware cleared, a PTR of type receipt and managed service name “Printer1”.

xfsMIBRoot.3.1.3.2	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceName)
	“Printer1”
xfsMIBRoot.3.1.3.3	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceClass)
	1 (WFS_SERVICE_CLASS_PTR)
xfsMIBRoot.3.1.3.4	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceClassName)
	“PTR”
xfsMIBRoot.3.1.3.5	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceType)
	1 (WFS_PTR_TYPERECEIPT)
xfsMIBRoot.3.1.3.6	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapManagedServiceOid)
	“.1.3.6.1.4.1.16213.2.1”
xfsMIBRoot.3.1.3.7	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapPhysicalDeviceName)
	“ReceiptPrinter1”
xfsMIBRoot.3.1.3.8	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapDeviceVendor)
	“Best Printers Incorporated”
xfsMIBRoot.3.1.3.9	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapMIBVersion)
	“1.10”

xfsMIBRoot.3.1.3.10	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapEvent)
	107 (WFS_USRE_PTR_PAPERTHRESHOLD)
xfsMIBRoot.3.1.3.11	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapDate)
	“20/03/2003 15:40:53 -300”
xfsMIBRoot.3.1.3.12	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsCommonTrapVars.xfsCommonTrapSPVersion)
	“1.23”
xfsMIBRoot.3.1.4.1	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsThresholdTrapVars.xfsThresholdTrapThresholdSource)
	2 (WFS_PTR_PAPERUPPER)
xfsMIBRoot.3.1.4.2	(xfsMIBRoot.xfsTrap.xfsTrapV1.xfsThresholdTrapVars.xfsThresholdTrapThresholdStatus)
	1 (WFS_PTR_PAPERLOW)

3. XFS Registry Configuration

A network management application uses configuration information to define the relationship between the managed services, physical devices and the parts of the management data that have to be provided to the remote SNMP Manager. In particular, this information defines the data included in the *General* sub-tree of XFS MIB tree (See the section 2.1 General Information).

The configuration information defines the managed services and includes specific information about the managed service and the physical devices, e.g. the physical device name.

The configuration information for an XFS subsystem is stored in the XFS configuration registry. The HKEY_LOCAL_MACHINE\SOFTWARE\XFS\MANAGEMENT_PROVIDERS key is the root key for all the XFS management configuration. Under this key are specific configuration values common to the XFS management system (ie not related to a specific managed service) and a key for each managed service.

3.1 XFS Common Management Configuration

The values defined within the root configuration key contain configuration information that applies to the overall XFS Management system or is common across all managed services. The following values are defined.

- RemoteDeviceResetAllowed — a flag indicating if all applications are able to co-operate with the SNMP Agent to control exclusive access to devices. If the value is '0' then the Device Reset is not allowed. If the value is '1' then Device Reset is allowed. If the value is not present or has any other value then Device Reset is not allowed.

The management provider configuration information for an example XFS system is shown below, where Device Reset is allowed and exclusive access to the device can be negotiated between the application and the SNMP agent.

HKEY_LOCAL_MACHINE\SOFTWARE\XFS Second Level Keys Third Level Keys (or values) Values
HKEY_LOCAL_MACHINE\SOFTWARE\XFS MANAGEMENT_PROVIDERS RemoteDeviceResetAllowed = '1'

3.2 Managed Services Configuration

There is one managed service for every logical interface offered by a physical device. For example on a recycler with a CIM and CDM interface there would be 2 entries in the registry, one for each interface. Managed services are used as the primary MIB key instead of logical services, as logical services can be defined by applications and would lead to duplicate data in the MIB.

Typically, the MANAGEMENT_PROVIDERS section of the registry will be defined by the vendor of the service provider, although a management solution provider can define their own entries if necessary. The sub-keys of the MANAGEMENT_PROVIDERS area of the registry will often reflect the SERVICE_PROVIDERS area, with one managed service key for each service provider key.

Each key name is unique for the workstation and identifies the name of the managed service. It has the following mandatory values:

MibVersion	the MIB class version supported by the managed service.
Class	the service class of the managed interface (see the Service Class Definition Document for the standard values).

ServiceProvider	the name of the service provider that provides the implementation of the standard XFS functionality for a logical interface (the key name of the corresponding service provider key).
PhysicalDeviceName	the name or names of the physical devices. Multiple devices are comma separated.
OID	object identifier of the device class managed.
ResetDeviceMediaControl	defines what happens to media when a device reset is issued and specifies the value reported in the xfsXXXResetDeviceMediaControl variable. For the set of valid values for each device class refer to the enumerated type defined for the xfsXXXResetDeviceMediaControl variable in each device class.

The management provider configuration information for an example XFS system is shown below.

HKEY_LOCAL_MACHINE\SOFTWARE\XFS	**Second Level Keys**	**Third Level Keys (or values)**	Values
HKEY_LOCAL_MACHINE\SOFTWARE\XFS	**XFS_MANAGER**		
	LOGICAL_SERVICES		
		<LogicalServiceName>	
			class
			provider
		<LogicalServiceName>	
	SERVICE_PROVIDERS		
		<ServiceProviderName>	
			DllName
			vendor_name
			version
		<ServiceProviderName>	
	MANAGEMENT_PROVIDERS		
		<ManagedServiceName>	
			MibVersion
			Class
			ServiceProvider
			PhysicalDeviceName
			OID
			ResetDeviceMediaControl
		<ManagedServiceName>	

The example below shows a possible registry configuration for a Card Reader that has a single device class interface:

_HKEY_USERS\DEFAULT\XFS
Second Level Keys
Third Level Keys (or values)
Values

LOGICAL_SERVICES
<CardReader1>
class = “IDC”
provider = “IBM1234”

_HKEY_LOCAL_MACHINE\SOFTWARE\XFS
Second Level Keys
Third Level Keys (or values)
Values

HKEY_LOCAL_MACHINE\SOFTWARE\XFS
XFS_MANAGER
TraceFile = “C:\XFSTRACE.LOG”

SERVICE_PROVIDERS
<IBM1234>
DllName = “IBM1234.dll”
vendor_name = “XFS Solutions Provider”
version = “1.0.1”

MANAGEMENT_PROVIDERS
<ManagedIBM1234>

MibVersion = “1.0.0”
Class = “IDC”
ServiceProvider = “IBM1234”
PhysicalDeviceName = “IBMUSBDIP”
OID = “.1.3.6.1.4.1.16213.2.2”
ResetDeviceMediaControl = “2”

4. XFS Service Provider Interface Management Extensions

Most of the information that is instrumented through the XFS MIB is already available through the existing XFS Device Class interfaces, primarily through the WFS_INF_XXX_STATUS and WFS_INF_XXX_CAPABILITIES commands, where XXX represents the device class. However, some of the required data is not provided by the existing XFS interfaces. This section defines changes to existing Information commands and additional Information commands that must be implemented by a service provider to allow reporting of the full XFS MIB data.

The new XFS MIB commands are all specified as Information commands executed by WFSGetInfo and WFSAsyncGetInfo. They are specified as Information commands so that they can be executed immediately, will not be queued and will not be affected by locks.

4.1 WFS_INF_XXX_CAPABILITIES

The Capabilities command is used to report the device capability information. The XFS MIB adds the following additional information to the lpszExtra field to report that the service provider supports the management commands.

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 extendable by service providers. Each string is null-terminated, the whole list terminated with an additional null character. An empty list may be indicated by either a NULL pointer or a pointer to two consecutive null characters.

For the service providers that report all of the XFS MIB data, this parameter will contain the following: XFS_MIB_VERSION=<0xnnnnnnnn>, where nnnnnnnn is the ASCII representation of a hexadecimal value. XFS_MIB_VERSION identifies the highest version of the MIB specification that the service provider supports. The low-order word contains the version number, while the high-order word must be set to zero. In the low-order word, the low-order byte specifies the major version number and the high-order byte specifies the minor version number. Note: in order to allow intermediate minor revisions (e.g. between 1.10 and 1.20), the minor version number should always be expressed as two decimal digits, i.e., 1.10, 1.11, 1.20, etc. As an example MIB version 1.0 is represented as "XFS_MIB_VERSION=<0x00000001>

4.2 WFS_INF_MIB_GET_RESPONSE_COUNTS

Description This command reports the response counts for the service provider. The service provider must maintain these response counts persistently across re-boots.

This command can either be used to return a single response count for a specified command and response code, or return response counts for all valid command codes and response codes.

When multiple command code/response codes are reported, only valid response codes for the associated command code can be returned in the output parameter. Valid is defined as those response codes specified for a particular command code within the relevant XFS device class interface specification and the generic response codes defined below.

In addition to the device class specific response codes the following generic response codes can be requested and reported:

- WFS_SUCCESS
- WFS_ERR_HARDWARE_ERROR
- WFS_ERR_SOFTWARE_ERROR
- WFS_ERR_OUT_OF_MEMORY
- WFS_ERR_TIMEOUT
- WFS_ERR_UNSUPP_COMMAND
- WFS_ERR_UNSUPP_DATA

Input Param `LPWFSMIBRESPONSECOUNT lpGetResponseCount`

```
typedef struct _wfs_mib_response_count
    {
    DWORD           dwCommandCode;
    LONG            lResponseCode;
    DWORD           dwResponseCount;
    } WFSMIBRESPONSECOUNT, * LPWFSMIBRESPONSECOUNT;
```

dwCommandCode
Specifies the command code for which the response count is required. This value is a command code as defined in the XFS device class interface specification. If this value is zero, data for all command codes and all response codes is returned, in this case *lResponseCode* is ignored.

lResponseCode
Specifies the specific response code associated with dwCommandCode for which the count is required. This value is an XFS response code value as defined in the XFS device class interface specification , or one of the generic response codes listed above.

dwResponseCount
This parameter is ignored as an input parameter by this command.

If lpGetResponseCount is NULL then information on all command codes and response codes is returned in the output parameter.

Output Param `LPWFSMIBRESPONSECOUNT * lpGetResponseCount;`

lpGetResponseCount
Pointer to a null-terminated array of pointers to response code count structures:

```
typedef struct _wfs_mib_response_count
      {
      DWORD           dwCommandCode;
      LONG            lResponseCode;
      DWORD           dwResponseCount;
   } WFSMIBRESPONSECOUNT, * LPWFSMIBRESPONSECOUNT;
```

dwCommandCode
Specifies the command code to which the response count applies. This value is a command code as defined in the XFS device class interface specification .

lResponseCode
Specifies the response code to which the count applies. This value is an XFS response code value as defined in the XFS device class interface specification , or one of the generic response codes listed above.

dwResponseCount
The count of the number of times that the specified response code was returned for the specified command code since all the counts were reset.

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

Value	Meaning
WFS_ERR_INVALID_DATA	The input parameters dwCommandCode or lResponseCode are outside the specified range for the service provider.
WFS_ERR_UNSUPP_DATA	The input parameters are valid but the requested count is not maintained by the service provider
WFS_ERR_UNSUPP_CATEGORY	The command is not supported although the service provider recognises the command.
WFS_ERR_INVALID_CATEGORY	The command is not supported and the service provider does not recognise the command.

Comments None.

4.3 WFS_INF_MIB_SET_RESPONSE_COUNT

Description This command sets a single response count for the service provider. The response counts are persistent.

Input Param `LPWFSMIBRESPONSECOUNT lpSetResponseCount`

```
typedef struct _wfs_mib_response_count
    {
    DWORD           dwCommandCode;
    LONG            lResponseCode;
    DWORD           dwResponseCount;
    } WFSMIBRESPONSECOUNT, * LPWFSMIBRESPONSECOUNT;
```

dwCommandCode
Specifies the command code for which the response count is to be set. This value is a command code as defined in the XFS device class interface specification .

lResponseCode
Specifies the response code for which the count is to be set. This value is an XFS error code value as defined in the XFS device class interface specification , or one of the generic response codes listed in the WFS_INF_MIB_GET_RESPONSE_COUNTS command description.

dwResponseCount
Value to set the response count to.

Output Param None.

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

Value	Meaning
WFS_ERR_INVALID_DATA	The input parameters dwCommandCode or lResponseCode are outside the specified range for the service provider.
WFS_ERR_UNSUPP_DATA	The input parameters are valid but the requested count is not maintained by the service provider
WFS_ERR_UNSUPP_CATEGORY	The command is not supported although the service provider recognises the command.
WFS_ERR_INVALID_CATEGORY	The command is not supported and the service provider does not recognise the command.

Comments None.

4.4 WFS_INF_MIB_RESET_RESPONSE_COUNTS

Description This command retrieves the timestamp when the response counts were reset and allows the response counts to be reset to zero.

Input Param `LPBOOL lpResetResponseCounts`

lpResetResponseCounts
Specifies if the response counts should be reset. If lpResetResponseCounts is TRUE then the all response counts are reset to zero and the timestamp of this reset is reported. If lpResetResponseCounts is FALSE then the counts are not reset but the timestamp of the last reset is reported.

Output Param `LPWFSMIBRESETRESPONSECOUNTS lpResetResponseCounts`

```
typedef struct _wfs_mib_reset_response_counts
   {
   SYSTEMTIME      tsTimestamp;
   } WFSMIBRESETRESPONSECOUNTS, * LPWFSMIBRESETRESPONSECOUNTS;
```

tsTimestamp
Time the reset occurred (local time, in a Win32 SYSTEMTIME structure)

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

Value	Meaning
WFS_ERR_UNSUPP_CATEGORY	The command is not supported although the service provider recognises the command.
WFS_ERR_INVALID_CATEGORY	The command is not supported and the service provider does not recognise the command.

Comments None.

5. Appendix A - General MIB sub-tree

In the following paragraph there is a sample of the XFS General MIB sub-tree in SMIv2 and SMIv1 format. For simplicity it is supposed that the XFS MIB is under the enterprise tree with identification number 16213.

5.1 General MIB and Trap MIB in SMIv2 and SMIv1 format

The following object contains the xfsGeneralMIB.MIB file in SMIv2 format.

SMIv2\
xfsGeneralMIB.MIB

The following object contains the xfsGeneralMIB.MIB file in SMIv1 format.

SMIv1\
xfsGeneralMIB.MIB

The following text is the content of xfsGeneralMIB.MIB file in SMIv2 format

```
-- ****************************************************************************
--  XFS GENERAL MIB
--  Management Information Base for XFS
--
--  Step 0.5
--
--  The XFS number is 16213
--  The ASN.1 prefix to, and including the XFS is: .1.3.6.1.4.1.16213
--
-- ****************************************************************************

 XFSMIB DEFINITIONS ::= BEGIN

    IMPORTS
       enterprises, Integer32, OBJECT-TYPE, OBJECT-IDENTITY, NOTIFICATION-TYPE

           FROM SNMPv2-SMI
       DisplayString
           FROM SNMPv2-TC;


-- ********************************************************************
-- General #defines
-- ********************************************************************
    IxfsMIBDeviceStatus ::= INTEGER
       {
       xfsDevOnline(1),
       xfsDevOffline(2),
       xfsDevPowerOff(3),
       xfsDevNoDevice(4),
       xfsDevHWError(5),
       xfsDevUserError(6),
       xfsDevBusy(7)
       }


    xfsMIBRoot OBJECT IDENTIFIER ::= { enterprises 16213 }

    xfsGeneral OBJECT IDENTIFIER ::= { xfsMIBRoot 1 }


--
*******************************************************************************
--
-- The XFS             (1.3.6.1.4.1.16213)
--       General Group       (1.3.6.1.4.1.16213.1)
--
-- Implementation of the general group is mandatory for all agents
```

```
-- supporting the XFS MIB.
--
-- ********************************************************************************
    xfsMIBV1 OBJECT IDENTIFIER ::= { xfsGeneral 1 }


    xfsMIBRelease OBJECT-TYPE
       SYNTAX Integer32
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "The XFS MIB release supported by the agent."
       ::= { xfsMIBV1 1 }


    xfsXFSRelease OBJECT-TYPE
       SYNTAX Integer32
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "The XFS documentation release that the MIB corresponds to."
       ::= { xfsMIBV1 2 }


    xfsJXFSRelease OBJECT-TYPE
       SYNTAX Integer32
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "The J/XFS documentation release that the MIB corresponds to."
       ::= { xfsMIBV1 3 }


    xfsManagedServices OBJECT-TYPE
       SYNTAX Integer32
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "The number of managed services present."
       ::= { xfsMIBV1 4 }


    xfsManagedServiceTable OBJECT-TYPE
       SYNTAX SEQUENCE OF XfsManagedServiceEntry
       MAX-ACCESS not-accessible
       STATUS current
       DESCRIPTION
           "Table of the managed services installed on the system."
       ::= { xfsMIBV1 5 }


    xfsManagedServiceEntry OBJECT-TYPE
       SYNTAX XfsManagedServiceEntry
       MAX-ACCESS not-accessible
       STATUS current
       DESCRIPTION
           "The managed service table entry."
       INDEX { xfsManagedServiceName }
       ::= { xfsManagedServiceTable 1 }


    XfsManagedServiceEntry ::=
       SEQUENCE {
           xfsManagedServiceName
              DisplayString,
           xfsManagedServiceClass
              Integer32,
           xfsManagedServiceType
              Integer32,
           xfsManagedServiceOID
              DisplayString,
           xfsManagedServicePhysicalDeviceName
              DisplayString,
           xfsManagedServiceVendor
              DisplayString,
```

```
       xfsManagedServiceMIBClassRelease
          Integer32,
       xfsManagedServiceInstance
          Integer32
    }

xfsManagedServiceName OBJECT-TYPE
   SYNTAX DisplayString (SIZE (0..255))
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
       "The name of the managed service."
   ::= { xfsManagedServiceEntry 1 }


xfsManagedServiceClass OBJECT-TYPE
   SYNTAX Integer32
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
       "The identifier of the XFS service class for the managed service."
   ::= { xfsManagedServiceEntry 2 }


xfsManagedServiceType OBJECT-TYPE
   SYNTAX Integer32
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
       "The type identifier of the XFS service class for the managed
       service."
   ::= { xfsManagedServiceEntry 3 }


xfsManagedServiceOID OBJECT-TYPE
   SYNTAX DisplayString (SIZE (0..255))
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
       "The OID of the sub-tree defining the management information for
       this class of managed service."
   ::= { xfsManagedServiceEntry 4 }


xfsManagedServicePhysicalDeviceName OBJECT-TYPE
   SYNTAX DisplayString (SIZE (0..255))
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
       "The name of the physical device or devices associated with this
       managed service. If there is more than one device, the names are
       comma-separated."
   ::= { xfsManagedServiceEntry 5 }


xfsManagedServiceVendor OBJECT-TYPE
   SYNTAX DisplayString (SIZE (0..255))
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
       "The name of the service provider vendor."
   ::= { xfsManagedServiceEntry 6 }


xfsManagedServiceMIBClassRelease OBJECT-TYPE
   SYNTAX Integer32
   MAX-ACCESS read-only
   STATUS current
   DESCRIPTION
       "The XFS MIB class release supported by service provider."
   ::= { xfsManagedServiceEntry 7 }


xfsManagedServiceInstance OBJECT-TYPE
   SYNTAX Integer32
```

```
       MAX-ACCESS read-only
       STATUS current
       DESCRIPTION
           "An arbitrary identifier for the managed service, assigned by the
           agent."
       ::= { xfsManagedServiceEntry 8 }


    xfsManagedService OBJECT IDENTIFIER ::= { xfsMIBRoot 2 }


--
********************************************************************************
*
--
-- The XFS (1.3.6.1.4.1.16213)
-- Managed Service  (1.3.6.1.4.1.16213.2)
--
-- Implementation of the managed service is mandatory for all agents
-- supporting the XFS MIB.
--
--
********************************************************************************
*
    xfsPTR OBJECT IDENTIFIER ::= { xfsManagedService 1 }

    xfsIDC OBJECT IDENTIFIER ::= { xfsManagedService 2 }

    xfsCDM OBJECT IDENTIFIER ::= { xfsManagedService 3 }

    xfsPIN OBJECT IDENTIFIER ::= { xfsManagedService 4 }

    xfsCHK OBJECT IDENTIFIER ::= { xfsManagedService 5 }

    xfsDEP OBJECT IDENTIFIER ::= { xfsManagedService 6 }

    xfsTTU OBJECT IDENTIFIER ::= { xfsManagedService 7 }

    xfsSIU OBJECT IDENTIFIER ::= { xfsManagedService 8 }

    xfsVDM OBJECT IDENTIFIER ::= { xfsManagedService 9 }

    xfsCAM OBJECT IDENTIFIER ::= { xfsManagedService 10 }

    xfsALM OBJECT IDENTIFIER ::= { xfsManagedService 11 }

    xfsCEU OBJECT IDENTIFIER ::= { xfsManagedService 12 }

    xfsCIM OBJECT IDENTIFIER ::= { xfsManagedService 13 }

    xfsTrap OBJECT IDENTIFIER ::= { xfsMIBRoot 3 }

    xfsTrapV2 OBJECT-IDENTITY
       STATUS current
       DESCRIPTION
           "Root node for the converted TRAP-TYPEs."
       ::= { xfsTrap 0 }


    xfsDSCTrap NOTIFICATION-TYPE
       OBJECTS { xfsDSCTrapSysName, xfsDSCTrapManagedServiceName,
                 xfsDSCTrapManagedServiceClass, xfsDSCTrapManagedServiceClassName,
                 xfsDSCTrapManagedServiceType,
                 xfsDSCTrapManagedServiceOid, xfsDSCTrapPhysicalDeviceName,
                 xfsDSCTrapDeviceVendor, xfsDSCTrapMIBVersion, xfsDSCTrapEvent,
                 xfsDSCTrapDate, xfsDSCTrapSPVersion,
                 xfsDSCTrapManagedServiceStatus }
       STATUS current
       DESCRIPTION
           "This trap indicates a change in the summary status of a managed
           service. Refer to the status table of this managed service to
           obtain details of the status."
       ::= { xfsTrapV2 1 }
```

```
    xfsErrorTrap NOTIFICATION-TYPE
       OBJECTS { xfsErrorTrapSysName, xfsErrorTrapManagedServiceName,
                 xfsErrorTrapManagedServiceClass,
                 xfsErrorTrapManagedServiceClassName,
                 xfsErrorTrapManagedServiceType,
                 xfsErrorTrapManagedServiceOid, xfsErrorTrapPhysicalDeviceName,
                 xfsErrorTrapDeviceVendor, xfsErrorTrapMIBVersion,
                 xfsErrorTrapEvent,
                 xfsErrorTrapDate, xfsErrorTrapSPVersion,
                 xfsErrorTrapSuggestedAction, xfsErrorTrapDescrString,
                 xfsErrorTrapAppId
               }
       STATUS current
       DESCRIPTION
           "This trap indicates a hardware or software error. Refer to the
           status table of this managed service to obtain details of the
           status."
       ::= { xfsTrapV2 2 }

    xfsThresholdTrap NOTIFICATION-TYPE
       OBJECTS { xfsCommonTrapManagedServiceName, xfsCommonTrapManagedServiceClass,
                 xfsCommonTrapManagedServiceClassName,
                 xfsCommonTrapManagedServiceType, xfsCommonTrapManagedServiceOid,
                 xfsCommonTrapPhysicalDeviceName, xfsCommonTrapDeviceVendor,
                 xfsCommonTrapMIBVersion, xfsCommonTrapEvent, xfsCommonTrapDate,
                 xfsCommonTrapSPVersion, xfsThresholdTrapSource,
                 xfsThresholdTrapStatus }
       STATUS current
       DESCRIPTION
           "This trap exposes details of events belonging to the class USER_EVENTS,
defined by the CEN XFS Interface Specification. Such events are specific to each
service class and indicate that some form of user intervention is required."
       ::= { xfsTrapV2 3 }


--  Copy here all the device specific MIBs
--
*******************************************************************************
*
--
--  The xfsMIBRoot  (1.3.6.1.4.1.16213)
--  Trap Description  (1.3.6.1.4.1.16213.3)
--
--  Implementation of both traps is mandatory for all agents
--  supporting the XFS MIB.
--
--
*******************************************************************************
*
    xfsTrapV1 OBJECT IDENTIFIER ::= { xfsTrap 1 }


-- ****************************************************************
-- Device Status Change Trap
-- ****************************************************************
    xfsDSCTrapVars OBJECT IDENTIFIER ::= { xfsTrapV1 1 }

    xfsDSCTrapSysName OBJECT-TYPE
       SYNTAX DisplayString (SIZE (0..255))
       MAX-ACCESS not-accessible
       STATUS current
       DESCRIPTION
           "The name of the workstation hosting the managed service
           generating the alarm."
       ::= { xfsDSCTrapVars 1 }

    xfsDSCTrapManagedServiceName OBJECT-TYPE
       SYNTAX DisplayString (SIZE (0..255))
       MAX-ACCESS not-accessible
       STATUS current
       DESCRIPTION
           "The name of the managed service generating the alarm."
       ::= { xfsDSCTrapVars 2 }

    xfsDSCTrapManagedServiceClass OBJECT-TYPE
```

```
      SYNTAX Integer32
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The identifier of the XFS service class for the managed service
          generating the alarm."
      ::= { xfsDSCTrapVars 3 }

   xfsDSCTrapManagedServiceClassName OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The name of the XFS service class for the managed service
          generating the alarm."
      ::= { xfsDSCTrapVars 4 }

   xfsDSCTrapManagedServiceType OBJECT-TYPE
      SYNTAX Integer32
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The type identifier of the XFS service class for the managed
          service generating the alarm."
      ::= { xfsDSCTrapVars 5 }

   xfsDSCTrapManagedServiceOid OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The OID of the sub-tree defining the management information for
          this class of managed service."
      ::= { xfsDSCTrapVars 6 }

   xfsDSCTrapPhysicalDeviceName OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The name of the physical device or devices associated with the
          managed service generating the alarm. If there is more than one
          device, the names are comma-separated."
      ::= { xfsDSCTrapVars 7 }

   xfsDSCTrapDeviceVendor OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The name of the service provider vendor associated with the
          managed service generating the alarm."
      ::= { xfsDSCTrapVars 8 }

   xfsDSCTrapMIBVersion OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The XFS MIB release that this trap conforms to."
      ::= { xfsDSCTrapVars 9 }

   xfsDSCTrapEvent OBJECT-TYPE
      SYNTAX Integer32
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The XFS event ID of the event generating the alarm."
      ::= { xfsDSCTrapVars 10 }

   xfsDSCTrapDate OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
```

```
            "This variable represents the UTC and bias for local translation
            of the date and time when the event was generated."
        ::= { xfsDSCTrapVars 11 }

    xfsDSCTrapSPVersion OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The vendor-defined version of the service provider generating the
            alarm."
        ::= { xfsDSCTrapVars 12 }

    xfsDSCTrapManagedServiceStatus OBJECT-TYPE
        SYNTAX IxfsMIBDeviceStatus
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The status of the managed service, which corresponds to the
            dwState field in the event data from the service provider."
        ::= { xfsDSCTrapVars 13 }


-- ********************************************************************
-- Error Trap
-- ********************************************************************
    xfsErrorTrapVars OBJECT IDENTIFIER ::= { xfsTrapV1 2 }

    xfsErrorTrapSysName OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The name of the workstation hosting the managed service
            generating the alarm."
        ::= { xfsErrorTrapVars 1 }

    xfsErrorTrapManagedServiceName OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The name of the managed service generating the alarm."
        ::= { xfsErrorTrapVars 2 }

    xfsErrorTrapManagedServiceClass OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The identifier of the XFS service class for the managed service
            generating the alarm."
        ::= { xfsErrorTrapVars 3 }

    xfsErrorTrapManagedServiceClassName OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The name of the XFS service class for the managed service
            generating the alarm."
        ::= { xfsErrorTrapVars 4 }

    xfsErrorTrapManagedServiceType OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The type identifier of the XFS service class for the managed
            service generating the alarm."
        ::= { xfsErrorTrapVars 5 }

    xfsErrorTrapManagedServiceOid OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        MAX-ACCESS not-accessible
```

```
      STATUS current
      DESCRIPTION
          "The OID of the sub-tree defining the management information for
          this class of managed service."
      ::= { xfsErrorTrapVars 6 }

   xfsErrorTrapPhysicalDeviceName OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The name of the physical device or devices associated with the
          managed service generating the alarm. If there is more than one
          device, the names are comma-separated."
      ::= { xfsErrorTrapVars 7 }

   xfsErrorTrapDeviceVendor OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The name of the service provider vendor associated with the
          managed service generating the alarm."
      ::= { xfsErrorTrapVars 8 }

   xfsErrorTrapMIBVersion OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The XFS MIB release that this trap conforms to."
      ::= { xfsErrorTrapVars 9 }

   xfsErrorTrapEvent OBJECT-TYPE
      SYNTAX Integer32
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The XFS event ID of the event generating the alarm."
      ::= { xfsErrorTrapVars 10 }


   xfsErrorTrapDate OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "This variable represents the UTC and bias for local translation
          of the date and time when the event was generated."
      ::= { xfsErrorTrapVars 11 }

   xfsErrorTrapSPVersion OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The vendor-defined version of the service provider generating the
          alarm."
      ::= { xfsErrorTrapVars 12 }

   xfsErrorTrapSuggestedAction OBJECT-TYPE
      SYNTAX Integer32
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The suggested action required to handle the error, it corresponds
          to the dwAction field in the XFS event data from the service
          provider."
      ::= { xfsErrorTrapVars 13 }

   xfsErrorTrapDescrString OBJECT-TYPE
      SYNTAX OCTET STRING (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
```

```
            "A description of the alarm, defined by the vendor of the service
            provider generating the alarm."
        ::= { xfsErrorTrapVars 14 }

    xfsErrorTrapAppId OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The ID of the application associated with the XFS session
            generating the alarm."
        ::= { xfsErrorTrapVars 15 }


-- ********************************************************************
-- Common Trap Variables
-- ********************************************************************
    xfsCommonTrapVars OBJECT IDENTIFIER ::= { xfsTrapV1 3 }

    xfsCommonTrapSysName OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The name of the workstation hosting the managed service
            generating the alarm."
        ::= { xfsCommonTrapVars 1 }

    xfsCommonTrapManagedServiceName OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The name of the managed service generating the alarm."
        ::= { xfsCommonTrapVars 2 }

    xfsCommonTrapManagedServiceClass OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The identifier of the XFS service class for the managed service
            generating the alarm."
        ::= { xfsCommonTrapVars 3 }


    xfsCommonTrapManagedServiceClassName OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The name of the XFS service class for the managed service
            generating the alarm."
        ::= { xfsCommonTrapVars 4 }

    xfsCommonTrapManagedServiceType OBJECT-TYPE
        SYNTAX Integer32
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The type identifier of the XFS service class for the managed
            service generating the alarm."
        ::= { xfsCommonTrapVars 5 }

    xfsCommonTrapManagedServiceOid OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
        MAX-ACCESS not-accessible
        STATUS current
        DESCRIPTION
            "The OID of the sub-tree defining the management information for
            this class of managed service."
        ::= { xfsCommonTrapVars 6 }

    xfsCommonTrapPhysicalDeviceName OBJECT-TYPE
        SYNTAX DisplayString (SIZE (0..255))
```

```
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The name of the physical device or devices associated with the
          managed service generating the alarm. If there is more than one
          device, the names are comma-separated."
      ::= { xfsCommonTrapVars 7 }


   xfsCommonTrapDeviceVendor OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The name of the service provider vendor associated with the
          managed service generating the alarm."
      ::= { xfsCommonTrapVars 8 }

   xfsCommonTrapMIBVersion OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The XFS MIB release that this trap conforms to."
      ::= { xfsCommonTrapVars 9 }

   xfsCommonTrapEvent OBJECT-TYPE
      SYNTAX Integer32
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The XFS event ID of the event generating the alarm."
      ::= { xfsCommonTrapVars 10 }

   xfsCommonTrapDate OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "This variable represents the UTC and bias for local translation
          of the date and time when the event was generated."
      ::= { xfsCommonTrapVars 11 }

   xfsCommonTrapSPVersion OBJECT-TYPE
      SYNTAX DisplayString (SIZE (0..255))
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The vendor-defined version of the service provider generating the
          alarm."
      ::= { xfsCommonTrapVars 12 }


   xfsCommonTrapResetDeviceResult OBJECT-TYPE
      SYNTAX Integer32
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The result of the request to reset the device. The values have the
          following meaning:
          zero -     the reset was executed successfully ( however  the device may
                     not be operational, see the device
                     status fields).
          one -      the reset was rejected because exclusive access could not be
                     obtained.
          two -      the reset was rejected because Device Resets are disabled on
                     this terminal.
          negative - the reset request was executed but failed and the value
                     corresponds to the XFS error code."
      ::= { xfsCommonTrapVars 13 }


-- ********************************************************************
-- Threshold Trap & Associated Variables
-- ********************************************************************
```

```
   xfsThresholdTrapVars OBJECT IDENTIFIER ::= { xfsTrapV1 4 }

   xfsThresholdTrapSource OBJECT-TYPE
      SYNTAX Integer32
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The device source component (e.g. bin number) associated with the user
          event, which only applies to some user events. A value of zero indicates
          that the event data did not include this detail, either because it does
          not apply to this user event or because the service provider supports a
          version of the XFS standard that does not define this event data."
      ::= { xfsThresholdTrapVars 1 }


   xfsThresholdTrapStatus OBJECT-TYPE
      SYNTAX Integer32
      MAX-ACCESS not-accessible
      STATUS current
      DESCRIPTION
          "The current threshold state. The value corresponds to the threshold
          value in the event data from the service provider."
      ::= { xfsThresholdTrapVars 2 }


END
```

6. Appendix B - C-Header files

6.1 XFSMIB.H

xfsmib.h

```
/******************************************************************************
*                                                                             *
* xfsmpi.h      XFS - MIB functions, types, and definitions                   *
*                                                                             *
*               Version 1.00  -- 20/01/2004                                   *
*                                                                             *
******************************************************************************/

#ifndef __inc_xfsmib__h
#define __inc_xfsmib__h

#ifdef __cplusplus
extern "C" {
#endif

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

/****** Common ***************************************************************/

#include <windows.h>
#include <xfsapi.h>

/* XFS MIB Command codes */
#define     WFS_MIB_OFFSET                          (60000)
#define     WFS_INF_MIB_GET_RESPONSE_COUNTS         (WFS_MIB_OFFSET+1)
#define     WFS_INF_MIB_SET_RESPONSE_COUNT          (WFS_MIB_OFFSET+2)
#define     WFS_INF_MIB_RESET_RESPONSE_COUNTS       (WFS_MIB_OFFSET+3)

/* XFS MIB Count structures common across all devices */
typedef struct _wfs_mib_response_count
{
    DWORD    dwCommandCode;
    LONG     lResponseCode;
    DWORD    dwResponseCount;
} WFSMIBRESPONSECOUNT, * LPWFSMIBRESPONSECOUNT;
typedef struct _wfs_mib_reset_response_counts
{
    SYSTEMTIME    tsTimestamp;
} WFSMIBRESETRESPONSECOUNTS, * LPWFSMIBRESETRESPONSECOUNTS;



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

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

#endif  /* __inc_xfsmib__h */
```