



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

WORKSHOP AGREEMENT

CWA 14050-11

November 2000

ICS 35.200; 35.240.40

Extensions for Financial Services (XFS) interface specification -
Release 3.0 - Part 11: Vendor Dependent Mode Class Interface

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

© 2000 CEN

All rights of exploitation in any form and by any means reserved world-wide for CEN National Members

Ref. No CWA 14050-11:2000 E

Table of Contents

Foreword	3
1. Introduction	5
1.1 Background to Release 3.0	5
1.2 XFS Service-Specific Programming.....	5
2. Vendor Dependent Mode	7
3. References	14
4. Info Commands	15
4.1 WFS_INF_VDM_STATUS.....	15
4.2 WFS_INF_VDM_CAPABILITIES	16
5. Execute Commands	17
5.1 WFS_CMD_VDM_ENTER_MODE_REQ.....	17
5.2 WFS_CMD_VDM_ENTER_MODE_ACK.....	17
5.3 WFS_CMD_VDM_EXIT_MODE_REQ	17
5.4 WFS_CMD_VDM_EXIT_MODE_ACK.....	18
6. Events	19
6.1 WFS_SRVE_VDM_ENTER_MODE_REQ	19
6.2 WFS_SRVE_VDM_EXIT_MODE_REQ	19
6.3 WFS_SYSE_VDM_MODEENTERED	19
6.4 WFS_SYSE_VDM_MODEEXITED	19
7. C-Header file	20

Foreword

This CWA is revision 3.0 of the XFS interface specification.

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

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

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

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

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

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

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

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

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

Part 2: Service Classes Definition; Programmer's Reference

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Revision History:

2.00	November 11, 1996	Initial release
3.00	October 18, 2000	Updated for XFS version 3.00

1. Introduction

1.1 Background to Release 3.0

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

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

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

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

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

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

1.2 XFS Service-Specific Programming

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

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

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

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

- The requested capability is defined for the class of service providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability is *not* considered to be fundamental to the service. In this case, the service provider returns a successful completion, but does no operation. An example would be a request from an application to turn on a control indicator on a passbook printer; the service provider recognizes the command, but since the passbook printer it is managing does not include that indicator, the service provider does no operation and returns a successful completion to the application.
- The requested capability is defined for the class of service providers by the XFS specification, the particular vendor implementation of that service does not support it, and the unsupported capability *is* considered to be fundamental to the service. In this case, a WFS_UNSUPP_COMMAND error is returned to the calling application. An example would be a request from an application to a cash dispenser to dispense coins; the service provider recognizes the command but, since the cash dispenser it is managing dispenses only notes, returns this error.
- The requested capability is *not* defined for the class of service providers by the XFS specification. In this case, a WFS_ERR_INVALID_COMMAND error is returned to the calling application.

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

2. Vendor Dependent Mode

This specification describes the functionality of the services provided by the Vendor Dependent Mode (VDM) services under XFS, by defining the service-specific commands that can be issued, using the **WFSGetInfo**, **WFSAsyncGetInfo**, **WFSExecute** and **WFSAsyncExecute** functions.

In all device classes there needs to be some method of going into a vendor specific mode to allow for capabilities which go beyond the scope of the current XFS specifications. A typical usage of such a mode might be to handle some configuration or diagnostic type of function or perhaps perform some 'off-line' testing of the device. These functions are normally available on Self-Service devices in a mode traditionally referred to as Maintenance Mode or Supervisor Mode and usually require operator intervention. It is those vendor-specific functions not covered by (and not required to be covered by) XFS Service Providers that will be available once the device is in Vendor-Dependent mode.

This service provides the mechanism for switching to and from Vendor Dependent Mode. The VDM Service Provider can be seen as the central point through which all Enter and Exit VDM requests are synchronised.

Entry into, or exit from, Vendor Dependent Mode can be initiated either by an application or by the VDM Service Provider itself. If initiated by an application, then this application needs to issue the appropriate command to request entry or exit. If initiated by the VDM Service Provider i.e. some vendor dependent switch, then these request commands are in-appropriate and not issued.

Once the entry request has been made, all registered applications will be notified of the entry request by an event message. These applications must attempt to close all open sessions with XFS Service Providers as soon as possible and then issue an acknowledgement command to the VDM Service Provider when ready. Once all applications have acknowledged, the VDM Service Provider will issue event messages to these applications to indicate that the System is in Vendor Dependent Mode.

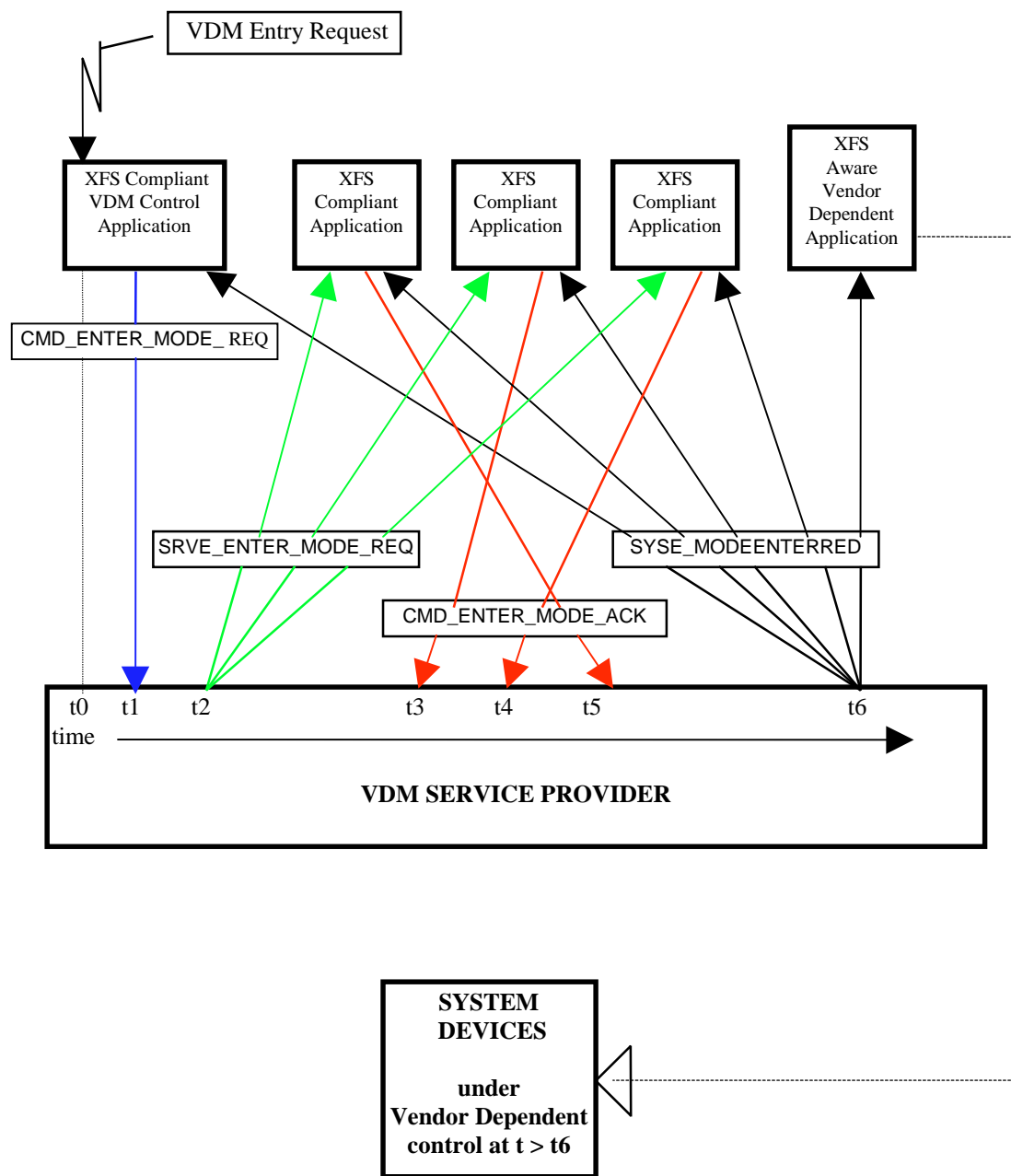
Similarly, once the exit request has been made all registered applications will be notified of the exit request by an event message. These applications must then issue an acknowledgement command to the VDM Service Provider immediately. Once all applications have acknowledged, the VDM Service Provider will issue event messages to these applications to indicate that the System has exited from Vendor Dependent Mode.

Thus, XFS compliant applications that do not need the system to be in Vendor Dependent Mode, must comply with the following:

- Every XFS application should open a session with the VDM ServiceProvider passing a valid ApplId and then register for all VDM entry and exit notices.
- Before opening any session with any other XFS Service Provider, check the status of the VDM Service Provider. If Vendor Dependent Mode is not "Inactive", do not open a session.
- When getting a VDM entry notice, close all open sessions with all other XFS Service Providers as soon as possible and issue an acknowledgement for the entry to VDM.
- When getting a VDM exit notice, acknowledge at once.
- When getting a VDM exited notice, re-open any required sessions with other XFS Service Providers.

This is mandatory for self-service but optional for branch.

VDM Entry triggered by XFS Application



At time t0, status is “Inactive” and a request to Enter VDM arises from within the Application system.

At time t1, an Application Process/Thread/Function issues the CMD_ENTER_MODE_REQ Execute cmd.

Status then becomes “Enter Pending”.

At time t2, the VDM Service Provider issues the SRVE_ENTER_MODE_REQ Event to all registered applications.

At time t3, the VDM Service Provider receives a CMD_ENTER_MODE_ACK Execute Command from a XFS Compliant Application

At time t4, the VDM Service Provider receives a CMD_ENTER_MODE_ACK Execute Command from another XFS Compliant Application

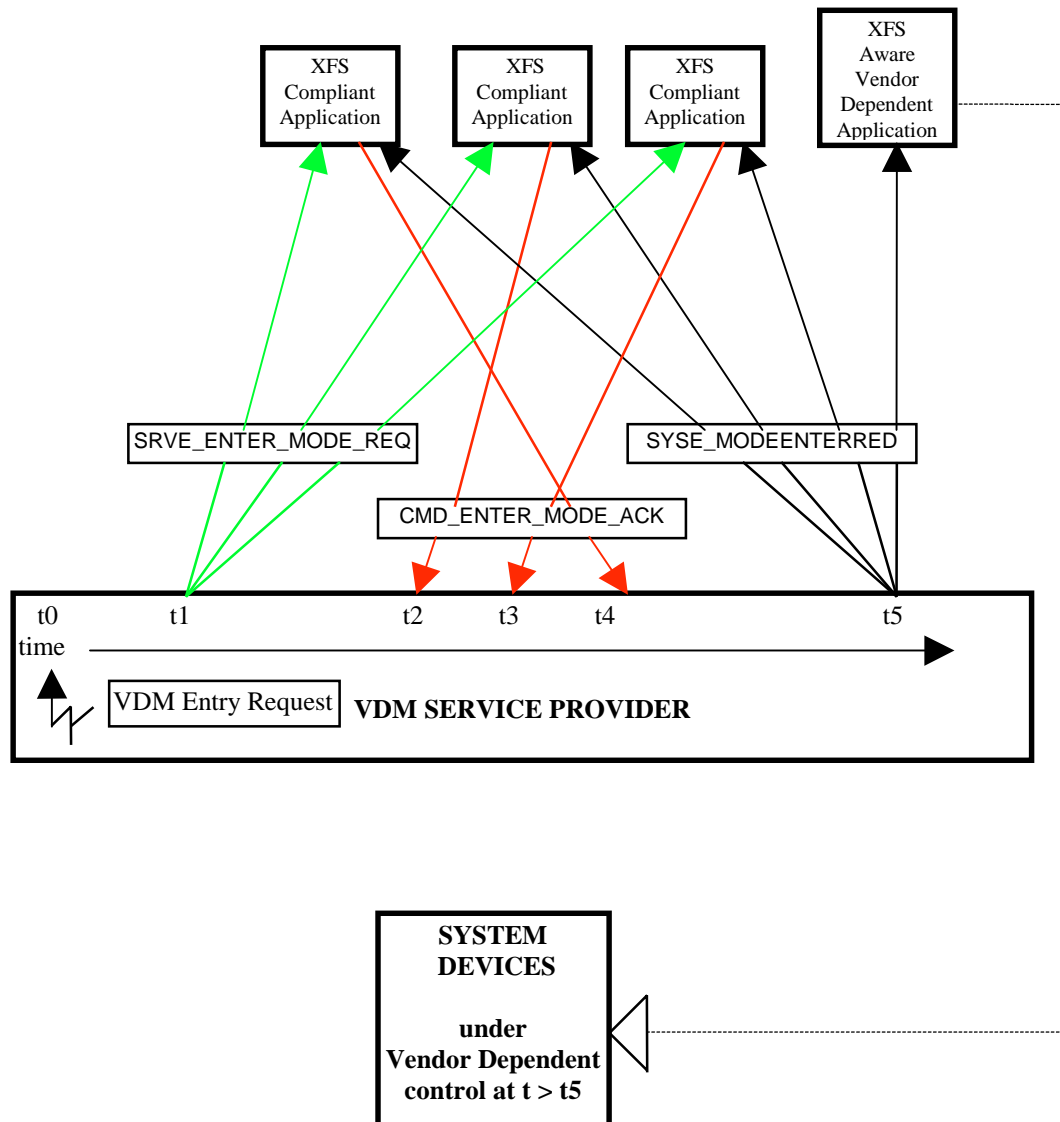
At time t5, the VDM Service Provider receives a CMD_ENTER_MODE_ACK Execute Command from the last XFS Compliant Application

At time t6, the VDM Service Provider issues the SYSE_MODEENTERED Event to all registered applications

Status then becomes “Active”.

The system is now in Vendor Dependent Mode and the Vendor Dependent Application can exclusively use the system devices in a Vendor Dependent manner.

VDM Entry triggered by Vendor Dependent Switch



At time t0, status is “Inactive” and a request to Enter VDM arises from within the Vendor System. Status then becomes “Enter Pending”.

At time t1, the VDM Service Provider issues the SRVE_ENTER_MODE_REQ Event to all registered applications.

At time t2, the VDM Service Provider receives a CMD_ENTER_MODE_ACK Execute Command from a XFS Compliant Application

At time t3, the VDM Service Provider receives a CMD_ENTER_MODE_ACK Execute Command from another XFS Compliant Application

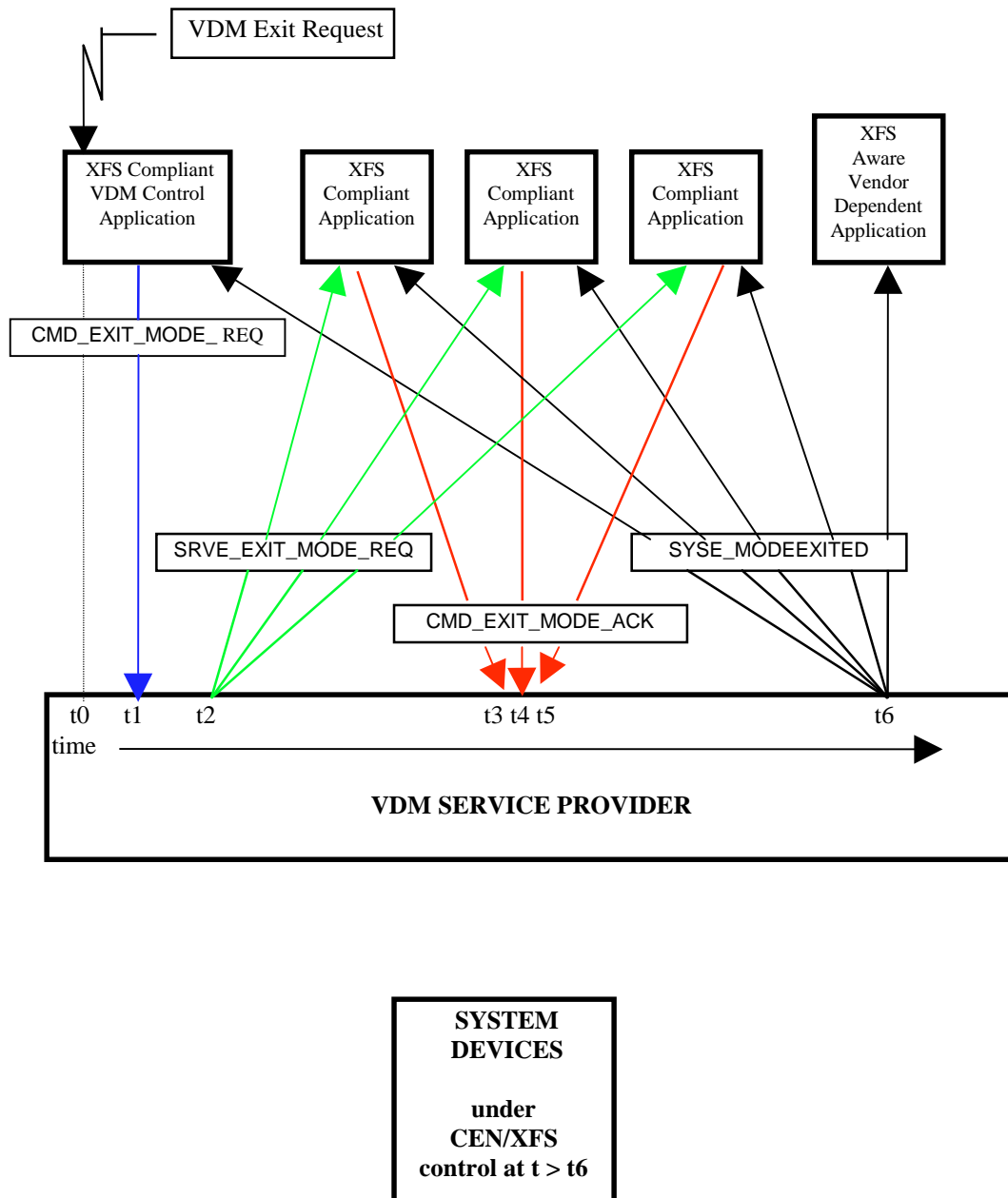
At time t4, the VDM Service Provider receives a CMD_ENTER_MODE_ACK Execute Command from the last XFS Compliant Application

At time t5, the VDM Service Provider issues the SYSE_MODEENTERRED Event to all registered applications

Status then becomes “Active”.

The system is now in Vendor Dependent Mode and the Vendor Dependent Application can exclusively use the system devices in a Vendor Dependent manner.

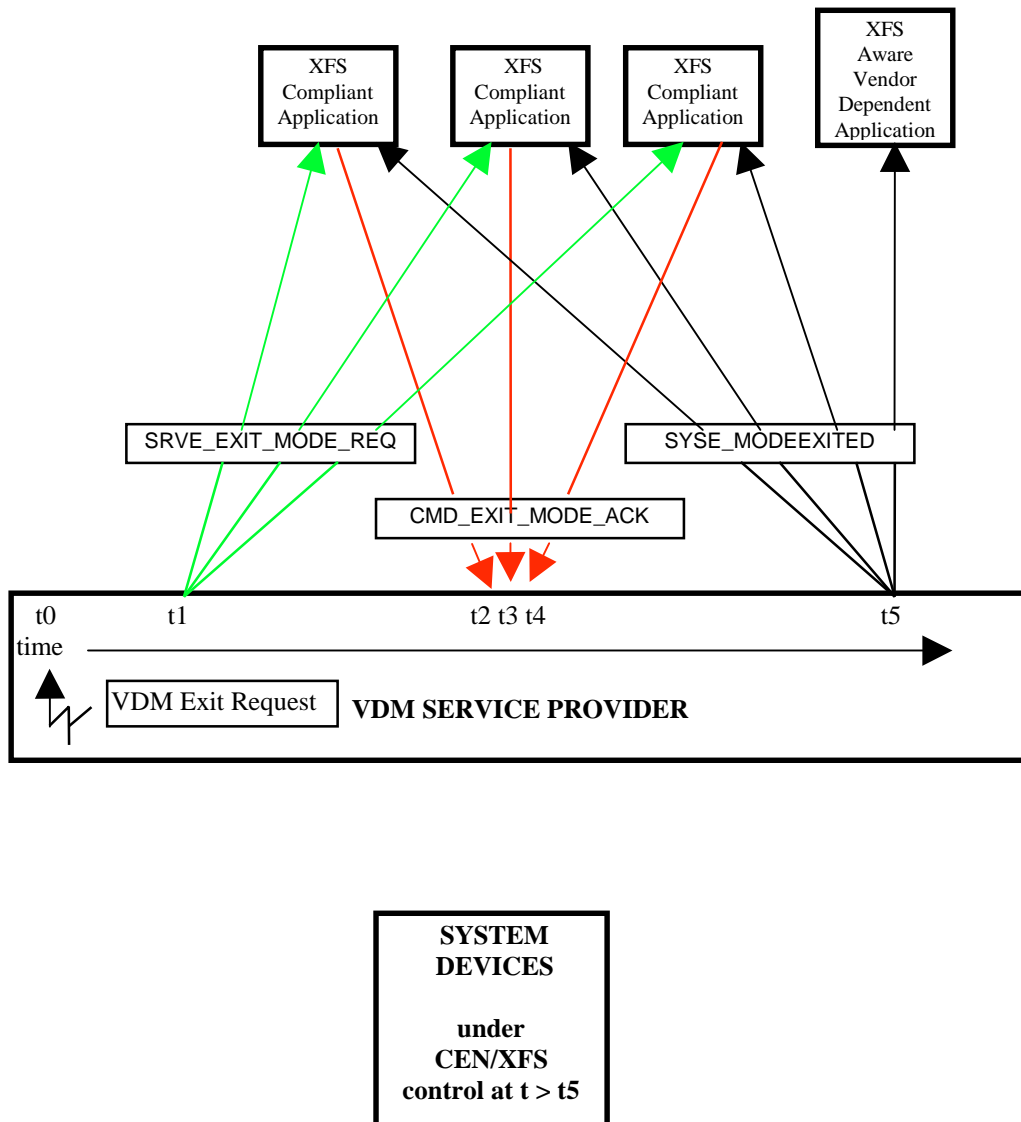
VDM Exit triggered by XFS Application



At time t0, status is “Active” and a request to Exit VDM arises from within the Application system.
 At time t1, an Application Process/Thread/Function issues the CMD_EXIT_MODE_REQ Execute cmd.
 Status then becomes “Exit Pending”.
 At time t2, the VDM Service Provider issues the SRVE_EXIT_MODE_REQ Event to all registered applications.
 At time t3, the VDM Service Provider receives a CMD_EXIT_MODE_ACK Execute Command from a XFS Compliant Application
 At time t4, the VDM Service Provider receives a CMD_EXIT_MODE_ACK Execute Command from another XFS Compliant Application
 At time t5, the VDM Service Provider receives a CMD_EXIT_MODE_ACK Execute Command from the last XFS Compliant Application
 At time t6, the VDM Service Provider issues the SYSE_MODEEXITED Event to all registered applications
 Status then becomes “Inactive”.

The system is now no longer in Vendor Dependent Mode and the XFS Compliant Applications can re-open any required services with other XFS Service Providers.

VDM Exit triggered by Vendor Dependent Switch



At time t0, status is “Active” and a request to Exit VDM arises from within the Vendor System. Status then becomes “Exit Pending”.

At time t1, the VDM Service Provider issues the SRVE_EXIT_MODE_REQ Event to all registered applications.

At time t2, the VDM Service Provider receives a CMD_EXIT_MODE_ACK Execute Command from a XFS Compliant Application

At time t3, the VDM Service Provider receives a CMD_EXIT_MODE_ACK Execute Command from another XFS Compliant Application

At time t4, the VDM Service Provider receives a CMD_EXIT_MODE_ACK Execute Command from the last XFS Compliant Application

At time t5, the VDM Service Provider issues the SYSE_MODEEXITED Event to all registered applications.

Status then becomes “Inactive”.

The system is now no longer in Vendor Dependent Mode and the XFS Compliant Applications can reopen any required services with other XFS Service Providers.

3. References

- | |
|--|
| <ol style="list-style-type: none">1. XFS Application Programming Interface (API)/Service Provider Interface (SPI), Programmer's Reference
Revision 3.00, October 18, 2000 |
|--|

4. Info Commands

4.1 WFS_INF_VDM_STATUS

Description This command indicates whether or not the system is in Vendor-Dependent Mode. It will also indicate which Applications have not responded to the VDM Request Enter Event or VDM Request Exit Event if the current Service status is WFS_VDM_ENTERPENDING or WFS_VDM_EXITPENDING respectively.

Input Param None.

Output Param LPWFSVDMSTATUS lpStatus;

```
typedef struct _wfs_vdm_status
{
    WORD                wDevice;
    WORD                wService;
    LPWFSVDMAPPSTATUS  * lppAppStatus;
    LPSTR               lpszExtra;
} WFSVDMSTATUS, * LPWFSVDMSTATUS;
```

wDevice

Specifies the status of the Vendor Dependent Mode Service Class. Status will be one of the following flags:

Value	Meaning
WFS_VDM_DEVONLINE	Vendor Dependent Mode service available.
WFS_VDM_DEVOFFLINE	Vendor Dependent Mode service unavailable.

wService

Specifies the Service state as one of the following flags:

Value	Meaning
WFS_VDM_ENTERPENDING	Vendor Dependent Mode Enter Request pending.
WFS_VDM_ACTIVE	Vendor Dependent Mode Active.
WFS_VDM_EXITPENDING	Vendor Dependent Mode Exit Request pending.
WFS_VDM_INACTIVE	Vendor Dependent Mode Inactive.

lppAppStatus

Pointer to a null-terminated array of pointers to Application Status structures:

```
typedef struct _wfs_vdm_appstatus
{
    LPSTR        lpszAppID;
    WORD        wAppStatus;
} WFSVDMAPPSTATUS, * LPWFSVDMAPPSTATUS;
```

lpszAppID

Application ID string

wAppStatus

Specifies whether the particular Application is ready for the System to enter or exit Vendor Dependent Mode. Values can be one of the following:

Value	Meaning
WFS_VDM_ENTERPENDING	Application not yet ready to Enter VDM.
WFS_VDM_ACTIVE	Application ready to Enter VDM.
WFS_VDM_EXITPENDING	Application not yet ready to Exit VDM.
WFS_VDM_INACTIVE	Application ready to Exit VDM.

lpszExtra

Specifies 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 will be null-terminated, with the final string terminating with two null characters.

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

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

4.2 WFS_INF_VDM_CAPABILITIES

Description This command is used to retrieve the capabilities of the VDM Service

Input Param None.

Output Param LPWFSVDMCAPS lpCaps;

```
typedef struct _wfs_vdm_caps
{
    WORD          wClass;
    LPSTR         lpSzExtra;
} WFSVDMCAPS, * LPWFSVDMCAPS;
```

wClass

Specifies the logical service class, value is:

WFS_SERVICE_CLASS_VDM

lpSzExtra

Specifies 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 will be null-terminated, with the final string terminating with two null characters.

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

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

5. Execute Commands

5.1 WFS_CMD_VDM_ENTER_MODE_REQ

Description This command is issued by an application to indicate a logical request to enter Vendor Dependent Mode. The VDM Service Provider will then indicate the request to all registered applications by sending a WFS_SRVE_VDM_ENTER_MODE_REQ Event message and then wait for an acknowledgement back from each registered application before putting the system into Vendor Dependent Mode. The Service Class status will change to WFS_VDM_ENTERPENDING on receipt of this command and will prevail until all applications have acknowledged, at which time the status will change to WFS_VDM_ACTIVE and the WFS_CMD_VDM_ENTER_MODE_REQ completes.

If the command fails when the status is WFS_VDM_ENTERPENDING, the status is changed to WFS_VDM_INACTIVE and a WFS_SYSE_VDM_MODEEXITED Event message is sent to all registered applications.

Input Param None.

Output Param None.

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

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

Value	Meaning
WFS_SRVE_VDM_ENTER_MODE_REQ	Request to enter VDM Mode.
WFS_SYSE_VDM_MODEENTERED	The system has entered VDM.
WFS_SYSE_VDM_MODEEXITED	The system has exited VDM.

Comments None.

5.2 WFS_CMD_VDM_ENTER_MODE_ACK

Description This command is issued by a registered application as an acknowledgement to the WFS_SRVE_VDM_ENTER_MODE_REQ Event message and it indicates that the application is ready for the system to enter Vendor Dependent Mode. Completion of this command is immediate.

Input Param None.

Output Param None.

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

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

Comments None.

5.3 WFS_CMD_VDM_EXIT_MODE_REQ

Description This command is issued by an application to indicate a logical request to exit Vendor Dependent Mode. The VDM Service Provider will then indicate the request to all registered applications by sending a WFS_SRVE_VDM_EXIT_MODE_REQ Event message and then wait for an acknowledgement back from each registered application before removing the system from Vendor Dependent Mode. The Service Class status will change to WFS_VDM_EXITPENDING on receipt of this command and will prevail until all applications have acknowledged, at which time the status will change to WFS_VDM_INACTIVE and the WFS_CMD_VDM_EXIT_MODE_REQ completes.

If the command fails when the status is WFS_VDM_EXITPENDING, the status is changed to WFS_VDM_ACTIVE and a WFS_SYSE_VDM_MODEENTERED Event message is sent to all registered applications.

Input Param None.

Output Param None.

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

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

Value	Meaning
WFS_SRVE_VDM_EXIT_MODE_REQ	Request to exit VDM Mode.
WFS_SYSE_VDM_MODEENTERED	The system has entered VDM.
WFS_SYSE_VDM_MODEEXITED	The system has exited VDM.

Comments None.

5.4 WFS_CMD_VDM_EXIT_MODE_ACK

Description This command is issued by a registered application as an acknowledgement to the WFS_SRVE_VDM_EXIT_MODE_REQ Event message and it indicates that the application is ready for the system to exit Vendor Dependent Mode. Completion of this command is immediate.

Input Param None.

Output Param None.

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

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

Comments None.

6. Events

6.1 WFS_SRVE_VDM_ENTER_MODE_REQ

Description This service event is used to indicate the request to enter Vendor-Dependent Mode.
Event Param None.
Comments None.

6.2 WFS_SRVE_VDM_EXIT_MODE_REQ

Description This service event is used to indicate the request to exit Vendor-Dependent Mode.
Event Param None.
Comments None.

6.3 WFS_SYSE_VDM_MODEENTERED

Description This system event is used to indicate that the system has entered Vendor-Dependent Mode.
Event Param None.
Comments None.

6.4 WFS_SYSE_VDM_MODEEXITED

Description This system event is used to indicate that the system has exited Vendor-Dependent Mode.
Event Param None.
Comments None.

7. C-Header file

```
/*
 *
 * xfsvdm.h XFS - Vendor Dependent Mode (VDM) definitions
 *
 *
 * Version 3.00 (10/18/00)
 *
 */
*****

#ifndef __INC_XFSVDM_H
#define __INC_XFSVDM_H

#ifdef __cplusplus
extern "C" {
#endif

#include <xfsapi.h>

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

/* values of WFSVDMCAPS.wClass */

#define WFS_SERVICE_CLASS_VDM (9)
#define WFS_SERVICE_CLASS_VERSION_VDM (0x0003) /* Version 3.00 */
#define WFS_SERVICE_CLASS_NAME_VDM "VDM"

#define VDM_SERVICE_OFFSET (WFS_SERVICE_CLASS_VDM * 100)

/* VDM Info Commands */

#define WFS_INF_VDM_STATUS (VDM_SERVICE_OFFSET + 1)
#define WFS_INF_VDM_CAPABILITIES (VDM_SERVICE_OFFSET + 2)

/* VDM Execute Commands */

#define WFS_CMD_VDM_ENTER_MODE_REQ (VDM_SERVICE_OFFSET + 1)
#define WFS_CMD_VDM_ENTER_MODE_ACK (VDM_SERVICE_OFFSET + 2)
#define WFS_CMD_VDM_EXIT_MODE_REQ (VDM_SERVICE_OFFSET + 3)
#define WFS_CMD_VDM_EXIT_MODE_ACK (VDM_SERVICE_OFFSET + 4)

/* VDM Messages */

#define WFS_SRVE_VDM_ENTER_MODE_REQ (VDM_SERVICE_OFFSET + 1)
#define WFS_SRVE_VDM_EXIT_MODE_REQ (VDM_SERVICE_OFFSET + 2)
#define WFS_SYSE_VDM_MODEENTERED (VDM_SERVICE_OFFSET + 3)
#define WFS_SYSE_VDM_MODEEXITED (VDM_SERVICE_OFFSET + 4)

/* values of WFSVDMSTATUS.wDevice */
#define WFS_VDM_DEVONLINE WFS_STAT_DEVONLINE
#define WFS_VDM_DEVOFFLINE WFS_STAT_DEVOFFLINE

/* values of WFSVDMSTATUS.wService */
#define WFS_VDM_ENTERPENDING (0)
#define WFS_VDM_ACTIVE (1)
#define WFS_VDM_EXITPENDING (2)
#define WFS_VDM_INACTIVE (3)

/*
 *
 * VDM Info Command Structures and variables
 *
 */
*****

typedef struct _wfs_vdm_appstatus
{
    LPSTR lpszAppID;
    WORD wAppStatus;
} WFSVDMAPPSTATUS, * LPWFSVDMAPPSTATUS;

typedef struct _wfs_vdm_status
{
    WORD wDevice;
    WORD wService;
}
```

```
        LPWFSVDMAPPSTATUS *lppAppStatus;
        LPSTR              lpszExtra;
} WFSVDMSTATUS, * LPWFSVDMSTATUS;

typedef struct _wfs_vdm_caps
{
        WORD                wClass;
        LPSTR              lpszExtra;
} WFSVDMCAPS, * LPWFSVDMCAPS;

/*=====*/
/* VDM Execute Command Structures */
/*=====*/

/*=====*/
/* VDM Message Structures */
/*=====*/

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

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

#endif /* __INC_XFSVDM__H */
```