C. & E. Becker
EDV-DIENSTLEISTUNGEN

Aschenbrödelweg 5
D-65199 Wiesbaden

✆ +49-611-2 94 95
carl@becker-wiesbaden.de

DOs AND DON'Ts
IN
MULTI-VENDOR CEN/XFS-DEVELOPMENT

Author : Carl Becker
Date : 26.5.2007

Preface

Today CEN/XFS is widely accepted in the ATM industry as an approach to allow ATM applications to run on a multitude of hardware platforms from different vendors. Therefore the development of multi-vendor applications by using CEN/XFS often is an explicit requirement from financial institutions.

However, purchasers and application developers have to be aware that the mere use of CEN/XFS to access the ATM hardware is a not a sufficient condition to make an application multi-vendor. Nor does the fact that an application is running smoothly on one type of ATM guarantee that it will do likewise on ATMs from a different vendor, not even on a different type of ATM from the same vendor.

The purpose of this paper is to give some advice to application developers about what to do and what to avoid in order to write multi-vendor applications.

Mind the physical differences between ATMs

Even though the purpose of CEN/XFS is to hide the differences between different devices from the application, there is some fundamentals that affect the business logic or the user interface of an ATM and thus must be handled by the application.

IDC Types

Motorized card readers, dip readers and swipe readers need different handling.

- motorized card readers need a call to WFS_CMD_IDC_EJECT
- dip readers read more reliable when the card is removed, so the consumer will be asked to remove the card after card insertion, whereas with motorized readers the card is usually held into the ATM until money is dispensed
- with a swipe reader the customer will be asked to pull through the card rather than insert and remove it

Operator Interface

If an application is known to run on frontload ATMs only, it can ignore any TTU device and use the customer screen for operating as well. This approach will not fit for rearload ATMs that usually have operating done with a TTU device.

Loading of Cryptographic Keys

There are various ways how cryptographic keys are loaded into an EPP

- keys entered in two parts on the operator panel and imported into the EPP
- keys entered in two parts directly into the EPP
- RKL (Remoe Key Loading) with signatures
- RKL with certificates
- No EPP is supporting all these key loading approaches[1] and financial institutions usually want just one of them to be used.

Missing devices

A card reader, a pinpad and a cash dispenser may be considered the minimum configuration for an ATM.

Depending on purchasers' requirements, additional devices may be built into an ATM such as a

- Journal printer
- Receipt printer
- Passbook printer
- Deposit unit
- Cash-In unit

[1] In fact this would be considered a security drawback

Applications should be prepared for these devices to be not installed and adjust their user interface if necessary[2].

Note that there is a difference between a device being

- not configured, which means it is not supposed to be existing
- configured but not responding due to a hardware problem[3]

IN CEN/XFS terms the former case is recognized either by Wfs[Async]Open() returning WFS_ERR_SERVICE_NOT_FOUND or the device status reporting fwDevice=WFS_XXX_DEVNODEVICE which is specified as

> There is no device intended to be there; e.g. this type of self service machine does not contain such a device or it is internally not configured.

whereas the latter case results in fwDevice being WFS_XXX_DEVPOWEROFF or WFS_XXX_DEVHWERROR.

It is up to the application's business logic whether it considers a missing device vital or continues operation without it.

Of course it makes application development easier if it can be restricted to certain ATM configurations. If, for example, the target configuration is front-load ATMs with motorized card readers and EPPs that support RKL it is perfectly acceptable for an application to not care about TTUs, dip readers and non-RKL ways of key distribution. However, everyone involved in a multi-vendor project has to be aware that this application will need modification in order to operate on other ATM configurations in future. The decision what configurations to support has to be made with care.

Check the Capabilities

A WFS_INF_XXX_CAPABILITES command is part of each CEN/XFS API for the various device classes. For some device classes there is additional GetInfo-commands that the application can use to find out details about the device[4]. Applications should adapt their behaviour to the specific device as far as possible.

The bells and whistles

ATMs may be equiped with a lot of optional widgets such as

- guidance lights
- lighting
- heating
- LEDs on the TTU

that are expected to be used if they are installed but don't really affect the ATM operation.

[2] e.g. if the receipt printer is missing, a receipt should not be offered to the customer nor should an attempt be made to print one

[3] such as a the power supply or the V24 line interrupted

[4] e.g. WFS_INF_PIN_FUNCKEY_DETAIL, WFS_INF_TTU_KEY_DETAIL

Even though some service providers may silently ignore attempts to set non-existing indicators with WFS_CMD_SIU_SET_PORT, others may return WFS_ERR_SIU_INVALID_PORT.

To be on the safe side an application should verify that an indicator is existing before using it.

Keys on the PIN and TTU keypad
The commands WFS_CMD_PIN_GET_DATA, WFS_CMD_PIN_GET_PIN and WFS_CMD_TTU_READ require the application to specifiy the active keys as part of the input parameter. Applications should make sure to specifiy keys that are actually existing only. The existing keys can be inquired by WFS_INF_PIN_FUNCKEY_DETAIL resp. WFS_INF_TTU_KEY_DETAIL.

Number and Location of FDKs
Applications should check the location of FDKs (softkeys) and adjust their screen layout accordingly.

Nowadays 8 softkeys (4 at each side of the screen) appears to be kind of standard for ATMs. However, developers should be prepared for ATMs that don't have any FDKs but rather are equiped with a touchscreen.

TTU layout
The capabilities of a TTU device include a list of supported display dimensions. There is no standard for the minumum rows and columns that have to be supported. In order to operate on different vendors' TTUs, applications probably have to restrict the dimension of TTU they require.

The number of FDKs on a TTU is obviously hardware-dependent as well.

The fewer FDKs and display rows / columns an application uses, the more likely it is to fit unchanged on different TTUs. Unfortunately here is some tradeoff to be made between usability and multi-vendor.

Safe Door
The sensor to detect the safe door state may be handled by the CDM, CIM and/or SIU service providers. Therefore an application should be prepared that any combination of WFS_SRVE_CDM_SAFEDOOROPEN, WFS_SRVE_CIM_SAFEDOOROPEN and WFS_SRVE_SIU_PORT_STATUS may be triggered if the safe door is opened.

Operator switch
Some ATMs have an operator switch or button that will trigger a WFS_SRVE_SIU_PORT_STATUS event when pressed to by service staff to request an operator session. These switches may have up to three states (WFS_SIU_RUN, WFS_SIU_MAINTENANCE and WFS_SIU_SUPERVISOR) but some have no distinction between the latter two states.

After the event has been triggered the state of fwSensors[WFS_SIU_OPERATORSWITCH] may immediately revert[5] to WFS_SIU_RUN or stay[6] at WFS_SIU_MAINTENANCE resp. WFS_SIU_SUPERVISOR. An application should be able to handle both buttons and switches.

The operator switch may be completely missing as well, in which case some other method must be used to start an operator session, such as pressing a key on the TTU or opening a cabinet door.

Avoid the use of vendor-dependent parts of the specification

There is features in the CEN/XFS specification that are explictely described as „vendor specific". Whenever possible, avoid the use of such features.

Raw Data Printing

The specification for WFS_CMD_PTR_RAW_DATA reads

> This command is used to send raw data (a byte string of device dependent data) to the physical device.

This command allows printer escape sequences to be sent by the application which is clearly not a multi-vendor approach.

lpszExtra

The specification for lpszExtra fields in the STATUS and CAPABILITIES commands reads:

> Pointer to a list of vendor-specific, or any other extended, information.

Even though the information provided here may be helpful for the application, developers must be aware that the use of such information reduces the applications's multi-vendor power.

Maintenance Commands

Some commands that are mainly intended for maintenance purposes, such as WFS_CMD_CDM_CALIBRATE_CASH_UNIT and WFS_CMD_CDM_TEST_CASH_UNITS are vendor specific. These commands are usually used by operator applications that in most cases need some knowledge about the hardware platform anyway.

WFS_CMD_PIN_GET_DATA

There has always been differences between implementations about the handling of keys in WFS_CMD_PIN_GET_DATA. In particular there is different implementations about what keys shall trigger events and what keys are to be included in the lpPinKeys parameter in the completion.

In order to avoid being affected by these differences an application should

- activate all existing keys
- set usMaxLen=0

[5] which is appropriate for an button that springs back to its normal position
[6] which is appropriate for an switch that catches in the operator position and needs to be reverted manually

- set bAutoEnd=false
- handle each keystroke as soon as the WFS_EXEE_PIN_KEY arrives and
- ignore the lpPinKeys field in the completion

Vendor-specific Extensions
Since the process of standardization in the CEN/XFS workshop is long and tedious whereas the market demands for additional features being implemented rather fast, time and again vendors have invented additional XFS commands or added new values to existing fields in the input or output parameters.

There is no guarantee that such vendor-specific extensions to the CEN/XFS API will ever become part of the standard. Even worse, vendor-specific command codes or parameter values may collide with future extensions to the standard.

The use of vendor-specific extensions should be restricted to cases where there is no other way to achieve the desired goal and the dangers that arise from using such extensions should clearly be communicated.

Don’t expect a specific behaviour to be generally valid

The CEN/XFS specification leaves some freedom to the implementor of a service provider. There is no guarantee that a specific behaviour observed with one device will be the same for all service providers.

Events and Completions
Applications shouldn’t rely on a specific sequence or timing of events.

For example, money can be taken before or after the WFS_CMD_CDM_PRESENT command completes. Some service providers may withhold the WFS_SRVE_CDM_ITEMSTAKEN event until after the completion of the command, others won’t.

The same applies for removal of ID cards and printed documents by the customer.

Since events are allowed to be sent while a command is being executed, an application must be able to handle them at whatever point they arrive.

Cancelling a Command

When cancelling a command with WfsAsyncCancel() there is no guarantee[7] that the command actually will be cancelled

- the command may be impossible to cancel[8]
- it may be too late to cancel the command[9]

The only way to detect about the outcome of a command is to wait for its completion.

WFSCDMCUINFO

Applications should not assume a specific number or order[10] of cash unit structures in the list of cash units returned by WFS_INF_CDM_CASH_UNIT_INFO.

Making assumptions about the currency or denomination is not a good idea either.

Cryptographic Keys

Applications should check with WFS_INF_PIN_KEY_DETAIL whether all the cryptographic keys they need are loaded. The fwEncStat field in the return from WFS_INF_PIN_STATUS is most likely not useful since the service provider cannot know what master keys the application actually needs.

Error Recovery

Applications should leave error recovery from hardware errors to the service provider who is closer to the hardware and knows better what to do in what error situation.

Immediately retrying an operation that has failed is in most cases not a good idea[11]. Applications should trust in the service provider that it has tried whatever is appropriate before completing a command with an error.

Since CEN/XFS 3.0 a WFS_CMD_XXX_RESET command is part of every device class API. Applications should use this command with caution, in particular no attempts should be made to reset a device in a loop.

Calling WfsClose() and WfsOpen() is not a way to do any reset operations.

Rebooting the whole ATM[12] may be a way to recover from errors. Again, applications should make sure they don't reboot in a loop when the ATM encounters the same error condition again and again after the reboot.

---

[7] In particular the fact that WfsAsncCancel() return WFS_SUCCESS tells nothing about the outcome of the command

[8] actually most commands are impossible to cancel and only those that wait for a human to perform an action such as pressing a key or inserting an ID card make sense to cancel

[9] e.g. in the very moment that the application decided to cancel the command, someone has inserted an ID card

[10] such as one reject, one retract cassette and four bill cassetes from top to bottom

[11] There had been cases where a card jammed and each of IDC firmware, XFS service and application retried the eject command 3 times, resulting in 27 attempts to eject the card

[12] by using the SIU service