PointTerminal

(APPROVED)

Document Name	: PointTerminal_2201.pdf
Document Type	: Technical Interface Specification
Version	: 02.200
Date	: 01-05-2006
Confidentiality	: NONE
By	: Jørgen Frits Hansen

Index

1. PointTerminal

This section describes the interface between Point payment terminals using PointTerminal. PointTerminal is an ActiveX control for application implementation in the Windows XP (32-bit) environment, developed and supported by Point A/S, and its main purpose is to simplify the interface to the SMASH/XENTA terminal, and still comply with the specifications in PBS OTRS 2.4.
PointTerminal ActiveX interface handles the certification requirements regarding:

- Populate and display terminal dialog-interfaces.
- Enabling the user to abort a transaction.
- Handles advanced issues like “verify signature” and “advice transfer flag”.
- Handles token transactions e.g. Original Authorization transaction
- Can optionally do the receipt printings.
- Implements an optional transaction preresult callback to ensure the application is alive.

PointTerminal includes a method that can call user selected administrative functions. In addition PointTerminal also include a management method that supports all administrative functions available in the terminal.

The objective for PointTerminal is to give the ECR access to do the basics:

- Make different types of transactions.
- Handle administrative functions e.g. to do the “end of day” settlement.
- Create reports

New since version 2.2.00 is printer control and translate code table possible.

2. Interface

The ActiveX control communicates with the user application using a dispatch map including a number of methods and an event interface.

2.1 Methods

2.1.1 SetConfiguration

SetConfiguration transfers required configuration parameters to the ActiveX Control. The user application must configure all mandatory parameters, before using other methods of the ActiveX control; otherwise the control process will fail. The parameters are outlined below, and they are explained in further details in subsequent section. **Bold** indicates default setting.
If the SetConfiguration function returns false (0), then the process logs the event and the parameters in the trace file.

NOTE: It is important that the SetConfiguration Trace option is called as the first SetConfiguration call in the start of the application that implements the ActiveX Control.

```
Method:      SetConfiguration  - configure


Prototype:   Long <ActiveX object_>SetConfiguration (
             Conf_Id As Long,
             parm1 As Long,
             parm2 As String,
             parm3 As String )

Input:       Conf_ID      - configuration id
             parm1        - configuration parameter
             parm2        - configuration parameter
             parm3        - configuration parameter

Return:      0 – configuration error
             1 – configuration success
```

Conf_ID.

0: Connection parameters.

Connection through Serial interface (RS232) or through TCP/IP (Ethernet). This parameter is **mandatory.**

parm1 = **1**: connect using RS232, 2: connect using = ethernet
parm2 = ComPort number or IP address
parm3 = Baudrate or TCP/IP Port

Examples:
<ActiveX object_>SetConfiguration(1,” 1”,”**9600**”) connect using RS232 seriel connection using **COM1**, 19.200 baud, 8N1.

<ActiceX object_>SetConfiguration(2,”192.168.0.53”,”2000”) connect to terminal on ip address 192.168.0.53 using tcp/ip protocol on port 2000.

1: Printer options.

Set options for printing if receipts. When enabled the ActiveX component do the receipt printing releasing the user application of this duty. The ActiveX supports a user defined translate and code table for simple control of pos printers. See appendix A.
When disabled the user application has to print the receipts. Normally this is done when a transaction is completed, and the receipt is available in the receipt file.
If the user application handles receipt printing, it must delete the print file after use. Otherwise the ActiveX component cannot start a new transaction.
If the ECR handles printing and a “verify signature” state is raised by PBS or the terminal, then the application will receive events signalling that receipts are ready to be printed. In this case the application must print the receipts, then delete the receipt file, and then set the event status to 1 for successful printing. This parameter is **optional.**

parm1 = **0**: auto print disabled, 1: auto print enabled
parm2 = Full path and filename of the receipt file, **“c:\bon.txt”**
parm3 = Full Windows Printer name as defined in printer settings, **“ScreenPrint”**

Example:
<ActiceX object_>SetConfiguration (1,0,”C:\MyApplicationDir\Bon.txt”,”\\networkprinter\HP_1000”)

2: Card options.

The user application will receive the first eight figures of the card number and the crc as an event during the transaction flow.
Set options for cards. When enabled the user application can accept (1) or reject the card number (0).
When disabled the ActiveX engine always replies accept to the terminal, and the cardnumber is saved in the specified card number file. This parameter is **optional.**

parm1 = 0: card confirm disabled, **1**= card confirm enabled.
parm2 = Full path and filename of card number file**, “c:\card.txt”**
parm3 = Not used

Example:
<ActiceX object_>SetConfiguration (2,1,”C:\card.txt”,””)

3: SetAmount.

Set options for transaction amount. If enabled the user application has the option to return a new transaction amount in event id 3. This event arrives after the card number is available giving the option to change the amount based on the card number and overrule the amount set in CardTransaction. This parameter is **optional.**

parm1 = **0**: set amount is disabled, 1: set amount is enabled.
parm2 = Not used
parm3 = Not used

Example:
<ActiveX object_>SetConfiguration(3,1,””,””)

4: TimeOutOK

Set dialog timeout of the transaction dialog. This parameter is **optional**. This parameter is **obsolete** and the dialog will automatically disappear **6** seconds if the transaction is approved. If not the dialog requires manually attention.

parm1 = **3**,4,5…x (seconds).
parm2 = Not used
parm3 = Not used

Example:
SetConfiguration (4,3,””,””) **OBSOLETE**

5: Trace

Set trace options. If enabled trace files are generated.
PointOcxTrace2200.txt – traces at the ActiveX level 1-3.
PtFxDrTr2000.txt – traces on the dll level 2-3.
flxComTrace.txt – traces the RS232/IP low level interface (if selected) 3.
Trace 4 enables traces – trace files is rewritten every time OpenTerminal is called.
Trace 5 enables traces appended to files.
This parameter is **optional**, but must be called as the first configuration parameter call if used.
Trace files are placed in the directory defined by printer options parm2.

parm1 = **0:** Trace is disabled,
1-3: Trace enabled,
4: Trace plus enabled,
5: Trace plus append enabled
parm2 = Not used
parm3 = Not used

Example:
SetConfiguration (5,1,””,””)

6: Get Amount Fee

Get amount fee options. If enabled the user application receives the fee calculated in the terminal in event number 6.

parm1 = **0:** Get amount fee is disabled, 1: get amount fee enabled.
parm2 = Not used
parm3 = Not used

Example:
SetConfiguration (6,1,"","")

7: Set Amount Fee

Set amount fee options. If enabled the user application can supply a transaction fee, overruling the fee generated by the terminal. The fee is set in event number 5. This parameter is optional.

parm1 = **0:** Set amount fee is disabled, 1: set amount fee enabled.
parm2 = Not used
parm3 = Not used

Example:
SetConfiguration (7,1,"","")

8: Get Extra Receipt Information

Get extra receipt information options. If enabled the user application receives extra receipt information at the end of a transaction. The information is received at event number 7. This parameter is optional. Extra receipt information is also received on signature receipts and receipt copies.

parm1 = **0:** Get info disabled, 1: get info enabled.
parm2 = Not used
parm3 = Not used

Example:
SetConfiguration (8,1,"","")

9: Get full card number and transaction reference

Get full card and transaction reference options. If enabled the user application receives full card and transaction reference. The information is received at event number 8. This parameter is optional.

parm1 = **0:** Get info disabled, 1: get info enabled.
parm2 = Not used
parm3 = Not used

Example:
SetConfiguration (9,1,"","")

10: Get dll timer event

Get dll timer event implements a callback from the lower level communication layer. This event number is 10 and is intended to avoid total blocking in case of communication errors. If used the application must close and disconnect the terminal before unloading the ocx object.

parm1 = **0:** Get timer event disabled, 1: get timer event enabled.
parm2 = Not used
parm3 = Not used

Example:
SetConfiguration (10,1,"","")

11: Set Amount Gratuity

Set amount gratuity options. If enabled the user application can supply a transaction gratuity. The gratuity is set in event number 11. This parameter is optional.

parm1 = **0:** Set amount gratuity is disabled, 1: set amount gratuity is enabled.
parm2 = Not used
parm3 = Not used

Example:
SetConfiguration (11,1,"","")

12: PreResult option.

PreResult is used to tell the terminal the user application is alive, this is done just before the receipt is going to be sent to the user application.

The user application will receive an event with the transaction result just before the receipt is going to be sent. The application must send a response (161 decimal) to this event to show it’s alive – otherwise the transaction will be rejected.

When disabled the ActiveX engine handles this internal. This parameter is **optional.**

parm1 = 0: preresult disabled, **1**= preresult enabled.
parm2 = Not used
parm3 = Not used

Example:
<ActiceX object_>SetConfiguration (12,1,””,””)

2.1.2 OpenTerminal

OpenTerminal connects the ActiveX control to the terminal and opens it, i.e. the terminal displays change from “Closed” to “Terminal Ready”.
Best practice is that the application keeps track of the connection status of the terminal, and issues the correct method sequence: OpenTerminal, CloseTerminal and DisconnectTerminal. This is important in error handling situations not to include these methods in program loops.
(If the terminal is not connected when the OpenTerminal function is called, then the PointTerminal automatically tries to connect to the terminal).

```
Method:      OpenTerminal      -

Prototype:   Long <ActiveX object_>OpenTerminal ()

Input:       -

Return:      0     Connect/open successful
             1     file error
             2     file read error
             3     Port initialising error
             4     Software not compatible
             5     Terminal not responding
             6     Com port already open error
             7     Datalink error (no connection)
             8     Internal function error
             9     Communication timeout
             10    Connect licence available in terminal
             11    Internal connect error
             19    Terminal Open but in ‘No receipt’ state

             255 - General internal OCX connect/open error
```

2.1.3 CloseTerminal

CloseTerminal closes the terminal, i.e. the terminal display change from “Terminal Ready” to “Welcome”.

```
Method:      CloseTerminal     -

Prototype:   Long <ActiveX object_>CloseTerminal ()

Input:       -

Return:      0  – Connect/close      successful
             1  – Connect/close error.
```

2.1.4 DisconnectTerminal

DisconnectTerminal disconnects the ActiveX control from the terminal, i.e. the terminal display change from “Welcome” to “Closed”.

```
Method:      DisconnectTerminal      -

Prototype:   Long <ActiveX object_>CloseTerminal ()

Input:       -

Return:      0  – Disconnect   successful
             1  – Disconnect   error.
```

2.1.5 CardTransaction

CardTransaction is the method to initiate and start a payment transaction.

```
Method:     CardTransaction   -


Prototype:  Long <ActiveX object_>CardTransaction (
            type As Long,
            amount As Long,
            currency As Long,
            ref As Long )

Input:      type        - transaction type
            amount      - transaction amount
            currency    - currency code
            ref         - user application reference number

Return:     0   – Transaction started successful
            11  – Receipt file exist
            12  – transaction type not valid
            254 – OCX state error
            255 – General OCX state error.
```

type:

The following transaction types are supported for the debit/credit applications. The OCX translates the type to defined transaction-type and merchant-initiative.
See the OTRS 2.4 specification for further descriptions of transaction-type, merchant-initiative and transaction tokens.

0: Default EMV Transaction
    Transaction-type = 0x00, merchant-initiative = 0x00

1: Forced PIN Transaction
    Transaction-type = 0x00, merchant-initiative = 0x81

2: Forced Signature Transaction
    Transaction-type = 0x00, merchant-initiative = 0x82

3: Refund Transaction
    Transaction-type = 0x01, merchant-initiative = 0x00

4: Offline Transaction
    Transaction-type = 0x00, merchant-initiative = 0x60

5: Original PIN Authorization Transaction
    Transaction-type = 0x02, merchant-initiative = 0x00

6: Original Signature Authorization Transaction
    Transaction-type = 0x02, merchant-initiative = 0x82

7: Supplemental Authorization Transaction
Transaction-type = 0x03, merchant-initiative defined by transaction token

8: Capture Transaction
Transaction-type = 0x04, merchant-initiative defined by transaction token

9: Reversal Transaction
Transaction-type = 0x05, merchant-initiative defined by transaction token

Amount:

Amount is always a positive value in the smallest currency unit (øre for DKK).

Currency:

ISO 4217 Currency code:

DKK	= 208	// DK Kroner
ISK	= 352	// IS Kroner
JPY	= 392	// JP Yen
NOK	= 578	// N Kroner
SEK	= 752	// S Kroner
CHF	= 756	// SW Francs
GBP	= 826	// GB Pound
USD	= 840	// US Dollar
EUR=978		// Euro

If the transaction is started successful the immediate return code is 0, then the final transaction result is returned in the CallBack event function with eventId 4.

NOTE: It is important that the implementation do not display dialogs during the transaction.

2.1.6 AbortTransaction

AbortTransaction is the method for the user application to abort a transaction previously started with the CardTransaction method. Only during the first part of the transaction flow, aborting is possible.

```
Method:      AbortTransaction  -

Prototype:   bool <ActiveX object_>AbortTransaction ()

Input:       -

Return:      true   – operator wish to abort
             false  – operator wish not to abort
```

2.1.7 Administration

Administration is the method for the user application to initiate administrative functions on the terminal. All the listed administrative functions are implemented in this method.
Functions 1, 8 and 20 are mandatory in the user application. The rest are optional.

```
Method:       Administration    -

Prototype:    Long <ActiveX object_>Administration (Function As Long)

Input:        function – administrative function

Return:       0 – function successful
              1 – function error
              65539 – Terminal in ‘No receipt’ state
```

Function:

1 :	**Do a advice transfer, also known as an ”end of day” transaction.**
2 :	Same as in function 1, but with a transaction log attached.
3 :	Get Terminal Report.
4 :	Get the current transaction totals sumarized per card- and currency type.
5 :	Get the current transaction log.
6 :	Get the previous transaction log.
7 :	Get last receipt from terminal.
8 :	**Get last receipt from terminal and unlock terminal.**
9 :	Do clock syncronisation towards PBS.
10 :	Do clock syncronisation towards Point.
11 :	Send the different logs to Point for further analysis.
12 :	**Clear (delete) the datastore. This action is irrevisible. Must be password protected.**
13 :	Download new program version – if any.
14 :	Download terminal configuration parameters.
15 :	Download ”local Card” Pan table.
16 :	Download TLCMDB –configuration settings.
17 :	Restore TLCMDB to the default settings.
18 :	Set contrast on PinPad (terminal) up
19 :	Set contrast on PinPad (terminal) download
20 :	**Restart terminal**
21-22:	Not used.
23 :	Turn on backlight on PinPad (terminal)
24 :	Turn off backlight on PinPad (terminal)

NOTE: It is important that the implementation do not display dialogs during an administrative procedure. If extra receipt information is enabled its event will include information for function 7 and 8, and zero for functions creating reports.

RECOMENDATION: Install all functions it’s a great help during instalation and in case of service on a terminal. Hide the more technical functions from the daily user, allowing only superusers, service users or tech. users … access to all.

2.1.8 SelectAdministration

The SelectAdministration method implements the selection of an administrative function listed above, as a dialog including a combo-box list that include all administrative functions. Implementing this function unlocks all administrative functions for the user.

Use the Administrative method if some administrative functions should remain locked (password protected) for the user.

```
Method:     SelectAdministration    -

Prototype:  Long <ActiveX object_>SelectAdministration ( )

Input:      -

Return:     0 -   no or nonexistent function selected
            1-20  Selected function as listed above.
```

RECOMANDATION: Always password protect this function, only superusers, service users or tech. users, should be allowed to do every administrative function.

2.1.9 CompleteExtCardTransaction

The CompleteExtCardTransaction method completes an external (local) card transaction.

```
Method:      CompleteExtCardTransaction    -

Prototype:  Long <ActiveX object_>CompleteExtCardTransaction(result As Long)

Input:      result - 0 approves the transaction
                     1 rejects the transaction

Return:     0 - method completed successful
            1 - method not completed successful
```

2.1.10 TransactionStatus

The TransactionStatus method returns the status of a previous transaction. This method is used in conjunction with the SetConfiguration id 9, Get full card number and transaction reference.
For terminals configured with PSAM version 50 and later, data of the last eight approved transactions are saved in the PSAM. The data of these transactions are retrieved using full card number, pan, or the unique transaction reference, stan, as search key.
The data are returned as event, CallBack, id 9.

```
Method:     TransactionStatus -

Prototype:  Long <ActiveX object_>TransactionStatus(keyT As bool, key As String)

Input:      keyT - 0 key is pan
                   1 key is stan

Return:     0 - method started successful
            1 – error
```

2.2 Event interface

The ActiveX component supply only one event interface.

2.2.1 CallBack

Communicates from the ActiveX component is done using an event interface.
The interface is a Call-back function labelled “CallBack”.
In the ActiveX control this function is presented as an event related to the control in the form.

```
Method:     CallBack    - event interface


Prototype:  Private Sub <ActiveX object_>CallBack (
            ByVal EventID As Long,
            ByVal Code As Long,
            ByVal Text As String,
            pResult As Long )

Input:      EventID     - unique event id
            Code        -
            Text        - variable contains any text related to the
                          specific event.
            pResult     - return value to <ActiveX object>  from user
                          application
```

Not all events are optional. In any case all events must give a return value.
An implementation using a select (or switch) statement is recommended.

EventID: (event number)

EventID 0: Date ready for print.

A receipt is ready to be printed, and placed in the specified file. The file must be removed after successful printing. If AutoPrint is selected then this event is not used.

Text contains a copy of the receipt, as in the specified file.

Code equal 0: – all receipts received, 1: - more receipts, 2: - receipt needs (signature) verification. Use bit 1 (0x00/0x01) for receipt cutting.

pResult is set to 1 if the receipt is accepted, and to 0 if not, in the latter case the terminal will be locked with status “NO RECIEPT” until unlocked.

EventID 1: For internal use.

pResult set to 1 by application.

EventID 2: Cardnumber available.

The event is used to return the cardtype, cardnumber and crc (card reconciliation code) to the user application. emv transactions only the first 8 numbers of the card number is available. The crc can be used to select a transaction fee.

If CardConfirm is configured to active, the application can accept or reject the card.

Text contains the cardnumber, crcnumber.

Code is 0 if the card is handled by PBS (cardtype). In the case pResult is set to 1 the cardnumber is accepted and the transaction is continued, if set to 0 it is aborted.

Code is 1 if the card is perceived as a "local card", and the intention is that the backend system does the subsequent transaction processing.

In this case **pResult** is set to 1 to accept the cardnumber. Note that in this case after the transactions is "completed the CompleteExtCardTransaction is must be called to finally completed the "local card" transaction. .

If **pResult** is set to 3 then cardholder is asked to confirm the amount on the terminal display, and the transaction is completed when this confirmation is done.

pResult set to 0 aborts the transaction.

EventID 3: Set Amount (changing amount after receiving cardnumber).

If Set Amount is configured to active, this event is used to return the amount of the transaction, by setting **pResult** to the amount. This amount value will overrule the value given in the cardtransaction function, and is mandatory if SetAmount is configured active.

EventID 4: Transaction result.

Result code of the transaction. Se also result codes for CardTransaction.
Text can contain a secondary result code.
Code holds the result code of the transaction.

0 : transaction successful
1 : transaction not successful
16: local card transaction started with success
17: local card transaction

text:
If present text is either a transaction return code from PBS, or a system specific return code from the driver. The PBS specific return codes is a long range of codes like a number for “Pincode Error”, “Stolen Card”, etc. These codes are not documented in this document – see the OTRS documentation for ASW1 ASW2 - and psam status codes.

System codes:

65536	Unknown error
65537	The terminal is not ready
65538	The connection is broken – unconnected
65539	No receipt. If code is 0 then the transaction was completed successfully, but the receipt is missing.
65540	Software is not compatible and the transaction can’t be started
65541	No licence available in terminal. Contact Point Customer Service

Remark…

As an option the PointTerminal can play a wav file in the transaction flow:

- SoundOK.wav will ”play” in case of a successful transaction.
- SoundError.wav will ”play” in case of a unsuccessful transaction.
- SoundAlert.wav will ”play” in case of question in the dialog during the transaction. As is the case of a ”advice transfer flag” or ”verify signature request”.

The sound files are optional, and only played if present in the same directory where the ocx is excecuted.

EventID 5: Set Amount Fee

If Set Amount Fee is configured to active, this event is used to return the fee amount of the transaction, by setting **pResult** to the fee amount. This amount value will overrule the fee amount calculated by the terminal. The function is mandatory if SetAmountFee is configured active.

EventID 6: Get Amount Fee

If Get Amount Fee is configured to active, this event returns the fee amount calculated by the terminal, to the user application. The fee amount is returned as an ASCII string in **text**, e.g. “50”.

EventID 7: Get extra receipt information

If Get Extra Receipt Info is configured to active, this event returns extra receipt information, to the user application. The information is returned as a comma separated string in text and includes the following tokens.

PAN,	- Full card number (primary account number)
Total,	- Total amount in smallest currency unit
Extra,	- Extra amount in smallest currency unit (not used)
Fee,	- Fee amount in smallest currency unit
Gratuity,	- Gratuity amount in smallest currency unit
Currency,	- ISO 4217 Currency code
STAN,	- Unique transaction reference number
PSAM,	- PSAM id
Action Code,	- Transaction result code
Cvm Status,	- Cardholder verification method see EMV specs
Card Source,	- Magnetic- or chip card
Aut. Code,	- PBS authorization code
Asw1Asw2	- PSAM return code
Card Name,	- Card name
Term Id,	- Terminal identification number
Number,	- Merchant PBS number
Name,	- Merchant name
City,	- Merchant city
Address,	- Merchant address
Zip,	- Merchant zip code
Phone,	- Merchant phone number
cvr,	- Merchant cvr number
Ref. Nr.	- User application reference number supplied with the CardTransaction method.

EventID 8: Full cardnumber and stan available

If get full Cardnumber and stan is configured to active, this event returns the full cardnumber and the unique transaction reference number, stan, as a comma separated ASCII string in **text**.

EventID 9: Transaction status

Result of the Transaction Status method. If **pResult** is 0 the transaction was found in the PSAM and transaction data are returned as a separated ASCII string in **text** with the following tokens:

stan	- unique transaction reference number, e.g. “000100”
amount	- amount, e.g. “1024”
currency	- currency code, e.g. “0208” for DKK
exponent	- currency exponent, e.g. “02” for DKK
timestamp	- transaction time stamp as DTHR, e.g. “0502281010” for 10:10, 28/2-2005

EventID 10: Dll timer event

If the Dll timer event is configured to active, this event is used to interrupt the ocx during timed functions in the low level interface to the terminal (dll).

EventID 11: Set Amount Gratuity

If Set Amount Gratuity is configured to active, this event is used to return the gratuity amount of the transaction, by setting **pResult** to the gratuity amount. The method is mandatory if SetAmountGratuity is configured active. Note that the terminal must be configured to use gratuity.

EventID 12: Get Transaction Token Reference

Some transactions output a transaction token e.g. performing an Original Authorization outputs a token that is needed when a Capture of the transaction is performed (for more information see OTRS 2.4). This method is mandatory if performing token transactions.
The OCX stores the Tokens on the disk in the directory defined by SetConfiguration, Printer options parm2 (default dir is C:\).
The file name of the token is made from the terminal id and the transaction reference number (stan), e.g. '0099075_000200.tok', from terminal id '0099075', transaction reference 000200. The transaction reference number is also printed on the transaction receipt.
Event ID 12 returns information about the token (a comma separated string).
CallBack pResult is set to 1.
It is the responsibility of the application to save this information for later retrieval of tokens.

Tokenid,	- unique reference to the token (transaction reference number (stan).
TokenFileName,	- the full filename of the saved token'
Amount,	- the original token amount
Currency	- the currency code of the original token amount

EventID 13: Put Transaction Token Reference

Performing transactions that require a token as input (the card need not be present) will generate this event ID 13. The TokenID (received in get Transaction Token event id 12, and converted to a Long) shall be returned in the CallBack pResult.
This method is mandatory if performing token transactions.

EventID 14: Put Authorization Code

Performing an offline transaction will require that an Authorization Code is returned in this event ID 14. The Authorization Code is issued by PBS, and a good practice is to get the code before the transaction is started.
The code is six characters (a-z, A-Z , 0-9 and space) and returned to the OCX by six successive CallBack with event ID 14, each returning one of the six character in pResult , starting with the most significant character. E.g if the code is 'TAST26' return 'T' in the first CallBack.
The method is mandatory if performing offline transactions.

EventID 15: Terminal ID

The OpenTerminal(…) function generates this event ID 15 with information about the connected terminal ID, an eight character string, eg. '00990075'.

EventID 16: PreResult status

Code is set to

0x00	Success
0x01	Success – signature/refund transaction - operator accepted signature
0x80	Transaction rejected by PBS/PSAM
0x81	Transaction rejected – signature/refund transaction – operator rejected signature
0x82	Transaction rejected – Other reasons

pResult must be set to 0xA1 (161 decimal) at all times to inform the terminal the user application is still alive. No matter the value of Code.

This method is mandatory if preresult is enabled.

EventID 255: Dialog Closing

If a dialog in the OCX was called, usually using the Administration method, this event ID 255 is generated when the dialog is closing, e.g. Cancel or OK is pressed

3 How to implement the ActiveX Control

3.1 ActiveX registration

To install PointTerimnal.ocx copy the file to an existing directory on your hard disk e.g. :
Chose Start Run and type cmd. In the command window type:

```
      copy PointTerminal.ocx c:\Program Files\PointWare\*.*
```

where `c:\Program Files\PointWare\` is the path of your choice.

Register the ActiveX control element in Windows by using the Regsvr32.exe program.

Chose Start Run and type cmd. In the command window type:

```
      Regsvr32 c:\program files\PointWare\PointTerminal.ocx
```

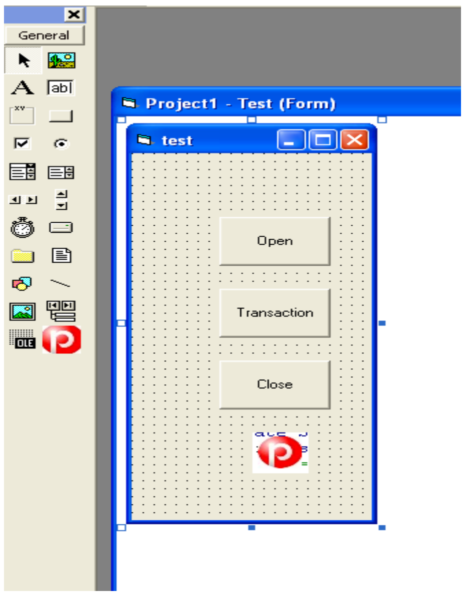
To unregister the OCX use

```
      Regsvr32 /u c:\program files\PointWare\PointTerminal.ocx
```

Consult the Windows documentation for more information about registrations.

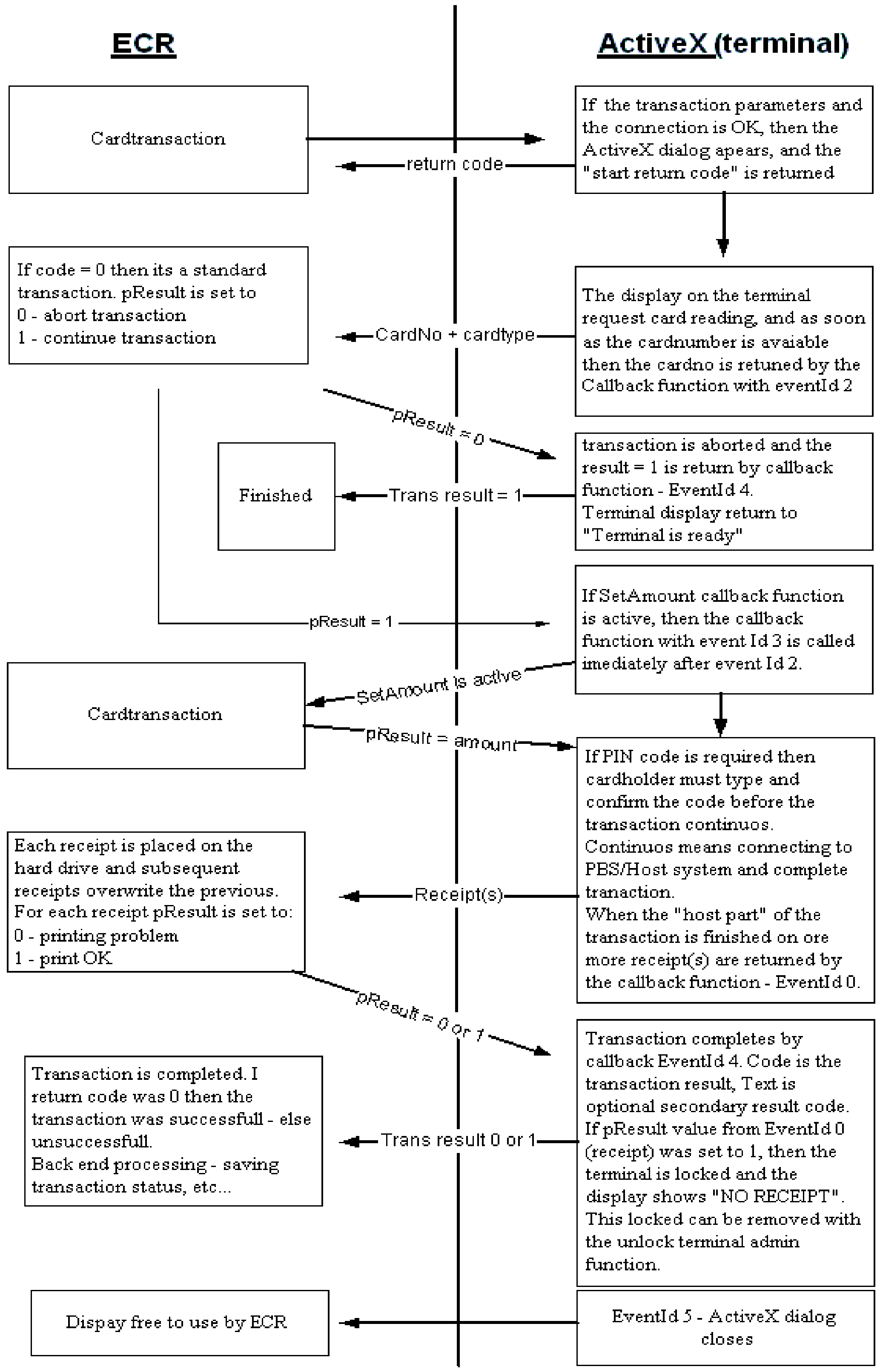
3.2 Including the ActiveX Control element in the development project

Add a reference to the ActiveX Control element in your application.
In MS Visual Basic select the Project menu, Components and check PointTerminal Active X Control module. The (P) icon will appear in the ToolBox.
This is illustrated in this figure showing the toolbar in Visual basic 6.0.



4. Flow

4.1 Transaction Flow



If the transaction parameters and the connection is OK, then the ActiveX dialog apears, and the "start return code" is returned

If code = 0 then its a standard transaction. pResult is set to
0 - abort transaction
1 - continue transaction

The display on the terminal request card reading, and as soon as the cardnumber is avaiable then the cardno is retuned by the Callback function with eventId 2

transaction is aborted and the result = 1 is return by callback function - EventId 4.
Terminal display return to "Terminal is ready"

If SetAmount callback function is active, then the callback function with event Id 3 is called imediately after event Id 2.

If PIN code is required then cardholder must type and confirm the code before the transaction continuos.
Continuos means connecting to PBS/Host system and complete tranaction.
When the "host part" of the transaction is finished on ore more receipt(s) are returned by the callback function - EventId 0.

Each receipt is placed on the hard drive and subsequent receipts overwrite the previous.
For each receipt pResult is set to:
0 - printing problem
1 - print OK

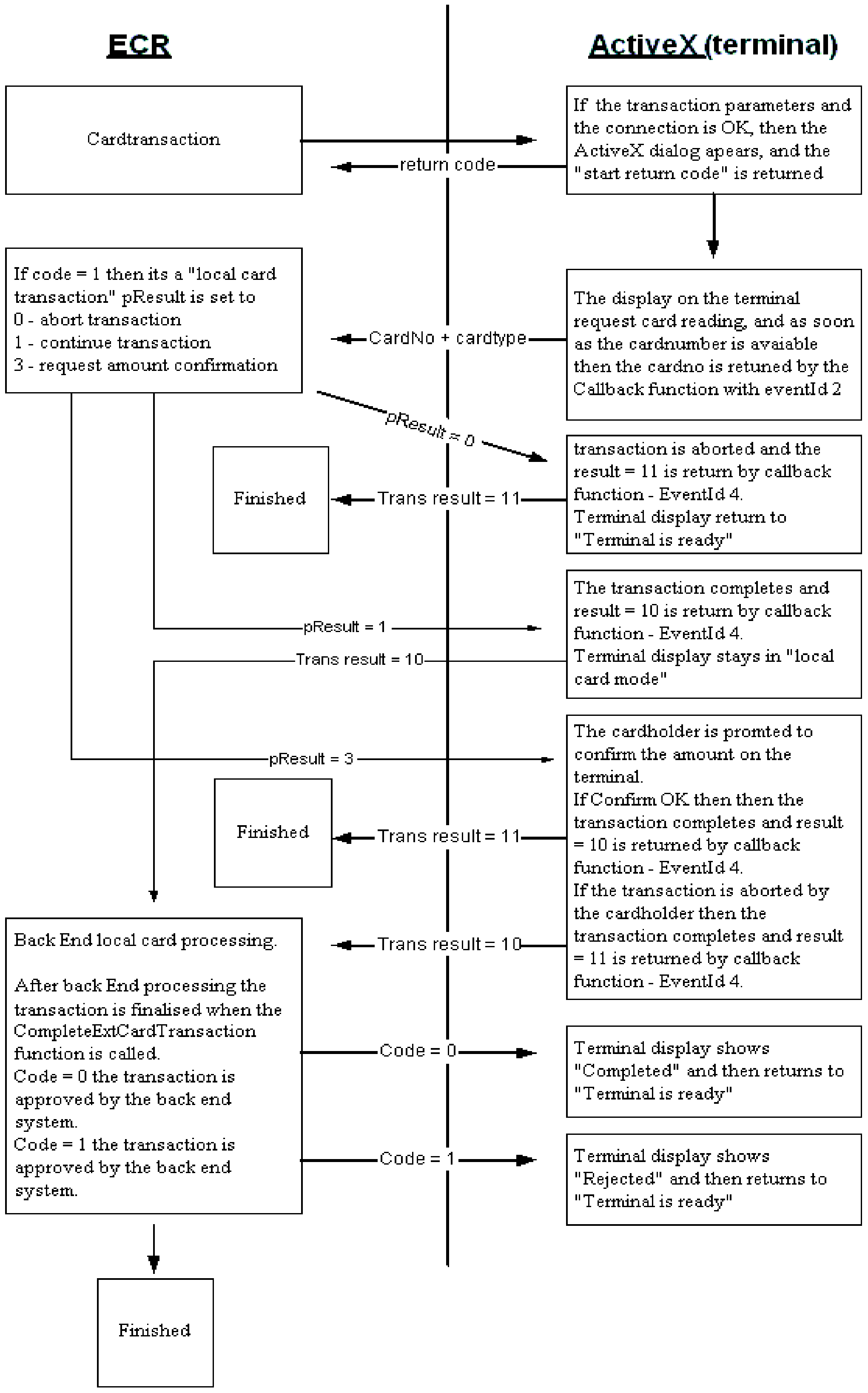
Transaction completes by callback EventId 4. Code is the transaction result, Text is optional secondary result code.
If pResult value from EventId 0 (receipt) was set to 1, then the terminal is locked and the display shows "NO RECEIPT".
This locked can be removed with the unlock terminal admin function.

Transaction is completed. I return code was 0 then the transaction was successfull - else unsuccessfull.
Back end processing - saving transaction status, etc...

Dispay free to use by ECR

EventId 5 - ActiveX dialog closes

4.2 External Card Transaction flow


ECR
ActiveX (terminal)
Cardtransaction
If the transaction parameters and the connection is OK, then the ActiveX dialog apears, and the "start return code" is returned
return code
If code = 1 then its a "local card transaction" pResult is set to
0 - abort transaction
1 - continue transaction
3 - request amount confirmation
CardNo + cardtype
The display on the terminal request card reading, and as soon as the cardnumber is avaiable then the cardno is retuned by the Callback function with eventId 2
pResult = 0
transaction is aborted and the result = 11 is return by callback function - EventId 4.
Terminal display return to "Terminal is ready"
Finished
Trans result = 11
pResult = 1
The transaction completes and result = 10 is return by callback function - EventId 4.
Terminal display stays in "local card mode"
Trans result = 10
pResult = 3
The cardholder is promted to confirm the amount on the terminal.
If Confirm OK then then the transaction completes and result = 10 is returned by callback function - EventId 4.
If the transaction is aborted by the cardholder then the transaction completes and result = 11 is returned by callback function - EventId 4.
Finished
Trans result = 11
Trans result = 10
Back End local card processing.
After back End processing the transaction is finalised when the CompleteExtCardTransaction function is called.
Code = 0 the transaction is approved by the back end system.
Code = 1 the transaction is approved by the back end system.
Code = 0
Terminal display shows "Completed" and then returns to "Terminal is ready"
Code = 1
Terminal display shows "Rejected" and then returns to "Terminal is ready"
Finished

4.3 External Card Transaction

A Local Card Transaction is an option available only if activated through a parameter download, meaning that the option requires a special setup in Points terminal Management system, and that these settings are downloaded into the terminal through a parameter download.
If a local.pan-file is present within a terminal, and a cardnumber matches on of the card ranges defined in the pan-file, then the transaction is handled as a ”local card transaction”.
As an extra option the cardholder can be promted to confirm the amount via the terminal display. This is the case if the user application must return 3 in the CallBack EventId.

APPENDIX A , printing

When configured to print the receipts, all output to the printer bypass the printer driver and is sent directly to the defined local or networked printer. The character encoding is according to ISO 8859-15.
A translation and command table is supported, intended for simplifying configuration of printers with simple control and character translation.
The table defines control commands (e.g. ESC/POS) send to the printer before and after the receipt.
The table is named ‘ExtendedAsciiPrint.txt’ and must be located in the same directory as your ECR application, (the application using the ActiveX).
The table entries define entries for indexes > 127 to value (in hex) and sends value to printer.
Format: <index> <value>[,<value>,…..] , where value is a two char hex value, 00 to FF.
If index is PRO or EPI the value (up to 127 bytes in hex and comma separated in one line) is send to the printer before (pro) and after (epi) the receipt data.

This example shows the table contents for an Epson TM-T88III printer.

```
PRO 1B,40,1B,74,10
EPI 1D,56,01,30
BC  8C       ; Œ
BD  9C       ; œ
```

The PRO part 1B,40 (ESC,@) initialises the printer and 1B,74,10 selects code table WPC1252.
The EPI part 1D,56,01,30 executes a partial cut of the receipt.
The character entries BC   8C etc. defines character translation.
Since WPC1252 is similar to ISO 8859-15 these entries are only shown as an example, and are not complete.