

## Re: SCardTransmit Fails

**Source:** <http://www.derkeiler.com/Newsgroups/microsoft.public.platformsdk.security/2004-03/0584.html>

---

**From:** Barry Hostead ([b.hostead\\_at\\_ntlworld.com](mailto:b.hostead_at_ntlworld.com))

**Date:** 03/25/04

Date: Thu, 25 Mar 2004 14:03:11 +0000

Pieter,

Thanks for your comments

I have attached the file that has all of the declarations in for this part of the code. I have tried re indexing the array but still no joy

Kind Regards

Barry

Pieter Philippaerts wrote:

> *"Barry Hostead" <[b.hostead@ntlworld.com](mailto:b.hostead@ntlworld.com)> wrote in message*

>

>> *Does anyone have any ideas my code for this section of the program is*

>> *listed below.*

>

>

> *Can you post the declarations you're using..?*

>

>

>> *SendBuff(1) = apduCommand.bCLA*

>> *SendBuff(2) = apduCommand.bINS*

>> *SendBuff(3) = apduCommand.bP1*

>> *SendBuff(4) = apduCommand.bP2*

>> *SendBuff(5) = apduCommand.bP3*

>

>

> *It's been a while since I've done VB.NET, but if I remember correctly VB.NET*

> *arrays start at index 0, not 1 [if I'm not mistaken, this was one of the*

> *changes between VB6 and VB.NET]. If this is the case, this may explain why*

> *your cann to SCardTransmit fails.*

>

> *Regards,*

> *Pieter Philippaerts*

> *Managed SSL/TLS: <http://mentalis.org/go.php?sl>*

>

>

```
Imports System.Text
Imports System.Runtime.InteropServices
Imports registrationModule.modcSideGlobal
```

```
Public Class clsPCSC
```

```
#Region "Comments"
```

```
*****
' This class is an adaptation of the "Winscard" API from Microsoft adapted by the
' Advanced Card Systems, Ltd for the Smart Card Development Kit sample code.
' The class was then further adapted by myself in order to use the class in
' Visual Basic .NET
'
```

```
' This class facilitates all of the functions required to connect the 'Smart Registration
' ~ ARMIS' to the ACR30U Smart Card Reader using the PC/SC Standard and Visual Basic.NET
```

```
*****
' Company : Advanced Card Systems, Ltd
' Author : Alcendor Lorzano Chan
' Creation Date : 18 / 09 / 2001
```

```
' Notes :
' Warnings :
*****
' Modified By : Barry John Hostead
' Modification Date : 09 / 12 / 2003
' Reasons for Modification : Module modified for use with VisualBasic.NET
```

```
#End Region
```

```
#Region "Type Declarations"
```

```
' I/O request control
Public Structure SCARD_IO_REQUEST
    ' Protocol identifier
    Public dwProtocol As Integer
    ' Protocol Control Information Length
    Public cbPciLength As Integer
End Structure
```

```
Public Structure APDURec
    Dim bCLA As Byte
    Dim bINS As Byte
    Dim bP1 As Byte
    Dim bP2 As Byte
    Dim bP3 As Byte
    Dim Data() As Byte
    Dim IsSend As Boolean
End Structure
```

```
#End Region
```

#Region "Public Variable and Constant Declarations"

'The following section is taken from the ACS Development Kit Sample Code,

'It includes all of the constant declarations

Dim ReturnCode As Long

Dim cardResponse As String

Public Const SCARD\_S\_SUCCESS = 0

Public Const SCARD\_ATR\_LENGTH = 33

'=====

' Context Scope

Public Const SCARD\_SCOPE\_USER = 0

' The context is a user context, and any  
' database operations are performed within the  
' domain of the user.

Public Const SCARD\_SCOPE\_TERMINAL = 1

' The context is that of the current terminal,  
' and any database operations are performed  
' within the domain of that terminal. (The  
' calling application must have appropriate  
' access permissions for any database actions.)

Public Const SCARD\_SCOPE\_SYSTEM = 2

' The context is the system context, and any  
' database operations are performed within the  
' domain of the system. (The calling  
' application must have appropriate access  
' permissions for any database actions.)

'=====

' Context Scope

Public Const SCARD\_STATE\_UNAWARE = &H0

' The application is unaware of the  
' current state, and would like to  
' know. The use of this value  
' results in an immediate return  
' from state transition monitoring  
' services. This is represented by  
' all bits set to zero.

Public Const SCARD\_STATE\_IGNORE = &H1

' The application requested that  
' this reader be ignored. No other  
' bits will be set.

Public Const SCARD\_STATE\_CHANGED = &H2

' This implies that there is a

' difference between the state  
' believed by the application, and  
' the state known by the Service  
' Manager. When this bit is set,  
' the application may assume a  
' significant state change has  
' occurred on this reader.

Public Const SCARD\_STATE\_UNKNOWN = &H4

' This implies that the given  
' reader name is not recognized by  
' the Service Manager. If this bit  
' is set, then SCARD\_STATE\_CHANGED  
' and SCARD\_STATE\_IGNORE will also  
' be set.

Public Const SCARD\_STATE\_UNAVAILABLE = &H8

' This implies that the actual  
' state of this reader is not  
' available. If this bit is set,  
' then all the following bits are  
' clear.

Public Const SCARD\_STATE\_EMPTY = &H10

' This implies that there is not  
' card in the reader. If this bit  
' is set, all the following bits  
' will be clear.

Public Const SCARD\_STATE\_PRESENT = &H20

' This implies that there is a card  
' in the reader.

Public Const SCARD\_STATE\_ATRMATCH = &H40

' This implies that there is a card  
' in the reader with an ATR  
' matching one of the target cards.  
' If this bit is set,  
' SCARD\_STATE\_PRESENT will also be  
' set. This bit is only returned  
' on the SCardLocateCard() service.

Public Const SCARD\_STATE\_EXCLUSIVE = &H80

' This implies that the card in the  
' reader is allocated for exclusive  
' use by another application. If  
' this bit is set,  
' SCARD\_STATE\_PRESENT will also be  
' set.

Public Const SCARD\_STATE\_INUSE = &H100

' This implies that the card in the  
' reader is in use by one or more  
' other applications, but may be  
' connected to in shared mode. If  
' this bit is set,  
' SCARD\_STATE\_PRESENT will also be  
' set.

Public Const SCARD\_STATE\_MUTE = &H200

' This implies that the card in the  
' reader is unresponsive or not  
' supported by the reader or  
' software.

Public Const SCARD\_STATE\_UNPOWERED = &H400

' This implies that the card in the  
' reader has not been powered up.

'=====  
'  
'=====

Public Const SCARD\_SHARE\_EXCLUSIVE = 1

' This application is not willing to share this  
' card with other applications.

Public Const SCARD\_SHARE\_SHARED = 2

' This application is willing to share this  
' card with other applications.

Public Const SCARD\_SHARE\_DIRECT = 3

' This application demands direct control of  
' the reader, so it is not available to other  
' applications.

'=====  
' Disposition  
'=====

Public Const SCARD\_LEAVE\_CARD = 0 ' Don't do anything special on close

Public Const SCARD\_RESET\_CARD = 1 ' Reset the card on close

Public Const SCARD\_UNPOWER\_CARD = 2 ' Power down the card on close

Public Const SCARD\_EJECT\_CARD = 3 ' Eject the card on close

'=====  
' Error Codes  
'=====

Public Const SCARD\_F\_INTERNAL\_ERROR = &H80100001

Public Const SCARD\_E\_CANCELLED = &H80100002

Public Const SCARD\_E\_INVALID\_HANDLE = &H80100003

Public Const SCARD\_E\_INVALID\_PARAMETER = &H80100004

Public Const SCARD\_E\_INVALID\_TARGET = &H80100005

```
Public Const SCARD_E_NO_MEMORY = &H80100006
Public Const SCARD_F_WAITED_TOO_LONG = &H80100007
Public Const SCARD_E_INSUFFICIENT_BUFFER = &H80100008
Public Const SCARD_E_UNKNOWN_READER = &H80100009
Public Const SCARD_E_TIMEOUT = &H8010000A
Public Const SCARD_E_SHARING_VIOLATION = &H8010000B
Public Const SCARD_E_NO_SMARTCARD = &H8010000C
Public Const SCARD_E_UNKNOWN_CARD = &H8010000D
Public Const SCARD_E_CANT_DISPOSE = &H8010000E
Public Const SCARD_E_PROTO_MISMATCH = &H8010000F
Public Const SCARD_E_NOT_READY = &H80100010
Public Const SCARD_E_INVALID_VALUE = &H80100011
Public Const SCARD_E_SYSTEM_CANCELLED = &H80100012
Public Const SCARD_F_COMM_ERROR = &H80100013
Public Const SCARD_F_UNKNOWN_ERROR = &H80100014
Public Const SCARD_E_INVALID_ATR = &H80100015
Public Const SCARD_E_NOT_TRANSACTED = &H80100016
Public Const SCARD_E_READER_UNAVAILABLE = &H80100017
Public Const SCARD_P_SHUTDOWN = &H80100018
Public Const SCARD_E_PCI_TOO_SMALL = &H80100019
Public Const SCARD_E_READER_UNSUPPORTED = &H8010001A
Public Const SCARD_E_DUPLICATE_READER = &H8010001B
Public Const SCARD_E_CARD_UNSUPPORTED = &H8010001C
Public Const SCARD_E_NO_SERVICE = &H8010001D
Public Const SCARD_E_SERVICE_STOPPED = &H8010001E
Public Const SCARD_W_UNSUPPORTED_CARD = &H80100065
Public Const SCARD_W_UNRESPONSIVE_CARD = &H80100066
Public Const SCARD_W_UNPOWERED_CARD = &H80100067
Public Const SCARD_W_RESET_CARD = &H80100068
Public Const SCARD_W_REMOVED_CARD = &H80100069
```

```
'=====
```

```
' Protocol
```

```
'=====
```

```
'Public Const SCARD_PROTOCOL_UNDEFINED = &H0 ' There is no active protocol.
```

```
'Public Const SCARD_PROTOCOL_T0 = &H1 ' T=0 is the active protocol.
```

```
'Public Const SCARD_PROTOCOL_T1 = &H2 ' T=1 is the active protocol.
```

```
Public Const SCARD_PROTOCOL_RAW = &H10000 ' Raw is the active protocol.
```

```
Public Const SCARD_PROTOCOL_DEFAULT = &H80000000 ' Use implicit PTS.
```

```
Public Const SCARD_PROTOCOL_UNDEFINED As Long = 0
```

```
Public Const SCARD_PROTOCOL_T0 As Long = 1
```

```
Public Const SCARD_PROTOCOL_T1 As Long = 2
```

```
'=====
```

```
' Reader State
```

```
'=====
```

```
Public Const SCARD_UNKNOWN = 0
```

```
' This value implies the driver is unaware
```

```
' of the current state of the reader.
```

```
Public Const SCARD_ABSENT = 1
```

```
' This value implies there is no card in the reader.
```

```
Public Const SCARD_PRESENT = 2
```

```
' This value implies there is a card is  
' present in the reader, but that it has  
' not been moved into position for use.
```

```
Public Const SCARD_SWALLOWED = 3
```

```
' This value implies there is a card in the  
' reader in position for use. The card is not powered.
```

```
Public Const SCARD_POWERED = 4
```

```
' This value implies there is power is  
' being provided to the card, but the  
' Reader Driver is unaware of the mode of the card.
```

```
Public Const SCARD_NEGOTIABLE = 5
```

```
' This value implies the card has been  
' reset and is awaiting PTS negotiation.
```

```
Public Const SCARD_SPECIFIC = 6 ' This value implies the card has been  
' reset and specific communication  
' protocols have been established.
```

```
'Friend defaultReader As String
```

```
#End Region
```

```
#Region "Private Variable and Constant Declaration"
```

```
Private iCount As Integer  
Private hContext As Integer  
Private hCard As Integer  
Private crdReaders As Collection  
Private crdProtocol As Integer  
Private strCrdReader As String
```

```
#End Region
```

```
#Region "Class winSCARD"
```

```
Public Class winSCard
```

```
'Establish communications with reader
```

```
Public Declare Ansi Function SCardEstablishContext Lib "winscard.dll" _
```

```
( _  
    ByVal dwScope As Integer, _  
    ByVal pvReserved1 As Integer, _  
    ByVal pvReserved2 As Integer, _  
    ByRef phContext As Integer _  
) As Integer
```

```
Public Declare Function SCardListReadersA Lib "winscard.dll" _
```

```
( _  
    ByVal hContext As Integer, _  
    ByVal mzGroup As String, _
```

```
    ByVal ReaderList As String, _  
    ByRef pchReaders As Integer _  
) As Integer
```

'Connect to a reader

```
Public Declare Function SCardConnectA Lib "winscard.dll" _  
( _  
    ByVal hContext As Integer, _  
    ByVal szReaderName As String, _  
    ByVal dwShareMode As Integer, _  
    ByVal dwPrefProtocol As Integer, _  
    ByRef hCard As Integer, _  
    ByRef ActiveProtocol As Integer _  
) As Integer
```

```
'Public Declare Function SCardReconnect Lib "winscard.dll" _  
( _  
' ByVal hCard As Integer, _  
' ByVal dwShareMode As Integer, _  
' ByVal dwPreferredProtocols As Integer, _  
' ByVal dwInitialization As Integer, _  
' ByRef pdwActiveProtocol As Integer _  
) As Integer
```

```
Public Declare Function SCardDisconnect Lib "Winscard.dll" _  
( _  
    ByVal hCard As Integer, _  
    ByVal Disposition As Integer _  
) As Integer
```

```
Public Declare Function SCardTransmit Lib "winscard.dll" _  
( _  
    ByVal hCard As Integer, _  
    ByVal pioSendRequest As SCARD_IO_REQUEST, _  
    ByRef SendBuff As Byte, _  
    ByVal SendBuffLen As Integer, _  
    ByVal pioRecvRequest As SCARD_IO_REQUEST, _  
    ByRef RecvBuff As Byte, _  
    ByRef RecvBuffLen As Integer _  
) As Integer
```

\*\*\*\*\*

```
Public Declare Function SCardReleaseContext Lib "Winscard.dll" _  
(ByVal hContext As Integer) As Integer
```

```
Public Declare Function SCardBeginTransaction _  
Lib "Winscard.dll" (ByVal hCard As Integer) As Integer
```

```
Public Declare Function SCardEndTransaction Lib "Winscard.dll" _  
(ByVal hCard As Integer, ByVal Disposition As Integer) As Integer
```

```
Public Declare Function SCardState Lib "Winscard.dll" _
(
    ByVal hCard As Integer, _
    ByRef State As Integer, _
    ByRef Protocol As Integer, _
    ByRef ATR As Byte, _
    ByRef ATRLen As Integer _
) As Integer

Public Declare Function SCardStatus Lib "Winscard.dll" Alias "SCardStatusA" _
(
    ByVal hCard As Integer, _
    ByVal szReaderName As String, _
    ByRef pcchReaderLen As Integer, _
    ByRef State As Integer, _
    ByRef Protocol As Integer, _
    ByRef ATR As Byte, _
    ByRef ATRLen As Integer _
) As Integer
End Class
#End Region

#Region "clsErrorHandler"
Public Class clsErrorMessage
    *****
    ' This class is an adaptation of one supplied by Axel Bessadale.
    ,
    *****

    *****

    ' Company :
    ' Author : Axel Bessadale
    ' Creation Date :
    ' Notes :
    ' Warnings :
    *****

    ' Modified By : Barry John Hostead
    ' Modification Date : 22 / 12 / 2003
    ' Reasons for Modification : Module modified for the purposes of error handling
    ' : within the Registration module of the
    ' : Smart Registration ~ ARMIS'
    *****
End Class

#Region "Declaration of Types and Functions"
Public Enum eFORMAT_MESSAGE
    'This Enumeration type declaration was supplied by
    'Axel Bessadale. It is for the purposes of formatting
    'The error message displayed to the user.
    FORMAT_MESSAGE_NONE = &H0
    FORMAT_MESSAGE_ALLOCATE_BUFFER = &H100
    FORMAT_MESSAGE_IGNORE_INSERTS = &H200
End Enum
```

```
FORMAT_MESSAGE_FROM_STRING = &H400
FORMAT_MESSAGE_FROM_HMODULE = &H800
FORMAT_MESSAGE_FROM_SYSTEM = &H1000
FORMAT_MESSAGE_ARGUMENT_ARRAY = &H2000
FORMAT_MESSAGE_MAX_WIDTH_MASK = &HFF
FORMAT_MESSAGE_ALL = &H1000 Or FORMAT_MESSAGE_IGNORE_INSERTS
End Enum
```

'This function was supplied by Axel Bessadale. It makes a call to the kernel32 library to display a more verbose error message to the user.

```
Private Declare Function formatMessage Lib "kernel32" Alias "FormatMessageA" _
```

```
( _
    ByVal dwFlags As Integer, _
    ByRef lpSource As Object, _
    ByVal dwMessageId As Integer, _
    ByVal dwLanguageId As Integer, _
    ByVal lpBuffer As String, _
    ByVal nSize As Integer, _
    ByRef Arguments As Integer _
) As Integer
```

```
#End Region
```

'The following code was supplied by Axel Bessadale and modified by Barry John Hostead.

'It is used to construct the output of the error message returning a property called

'errorMsg

```
Private errorMsg As String
```

```
ReadOnly Property errorMessage() As String
```

```
Get
```

```
Return errorMsg
```

```
End Get
```

```
End Property
```

```
Public Sub New(ByVal Buffer As Long)
```

```
Dim strErrorMessage As String = New String(CChar(" "), 999)
```

```
'Dim strMessage As New StringBuilder(999)
```

```
Dim ReturnCode As Integer
```

```
ReturnCode = formatMessage _
```

```
( _
    eFORMAT_MESSAGE.FORMAT_MESSAGE_ALL, _
    0, Buffer, 0, strErrorMessage, 999, 0 _
)
```

```
errorMsg = strErrorMessage
```

```
End Sub
```

```
End Class
```

```
#End Region
```

```
#Region "Access Functions and Subs"
```

```
*****
```

microsoft.public.platformsdk.security: Re: SCardTransmit Fails

' The following procedures functions and properties, have been adapted from a calss of  
' procedures provided by Axel ++++Bessadale++++, in response to a request for information  
' placed on ~ discuss.microsoft.com (Smart Card SDK Forum)

' I am very grateful to Axel for providing these procedures and resolving many problems.

\*\*\*\*\*

Friend ReadOnly Property Context() As Integer

    Get  
        Return hContext  
    End Get

End Property

Friend ReadOnly Property getCardReaders() As Collection

    'Property Supplied By Axel Bessadale

    Get  
        UpdateCardReaders()  
        Return crdReaders  
    End Get

End Property

ReadOnly Property GetProtocol() As Integer

    'Property Supplied By Axel Bessadale

    Get  
        Return crdProtocol  
    End Get

End Property

Property Reader() As String

    'Property Supplied By Axel Bessadale and Adapter by Barry John Hostead

    Get  
        Return strCrdReader  
    End Get

Set(ByVal Value As String)

    Dim ReaderFound As Boolean = False

    For iCount = 0 To getCardReaders.Count

        If getCardReaders.Item(iCount) = Value Then ReaderFound = True  
    Next

    If ReaderFound = False Then Throw New Exception( \_

        "Reader '" & strCrdReader & "'not found" \_

    ) : Exit Property

    strCrdReader = Value

    End Set

End Property

Private Sub establishContext()

    'Procedure Supplied By Axel Bessadale and Adapter by Barry John Hostead

    appendLog()

    ReturnCode = winSCard.SCardEstablishContext(SCARD\_SCOPE\_USER, 0, 0, hContext)

```
If ReturnCode <> SCARD_S_SUCCESS Then
    Dim cError As New clsErrorMessage(ReturnCode)
    Throw New Exception(cError.errMessage)
    fileWriter.WriteLine("Error Connecting to reader" & cError.errMessage)
Else
    fileWriter.WriteLine("SCardEstablishContext OK")
End If
closeLog()
End Sub

Private Sub UpdateCardReaders()
    'Procedure Supplied By Axel Bessadale and Adapter by Barry John Hostead
    Dim ReturnCode As Integer
    Dim strReaders As String = New String(CChar(" "), 999)
    Dim readerBufferLnth As Integer = 999
    Dim Readers() As String
    If hContext = 0 Then establishContext()
    Try
        ReturnCode = winSCard.SCardListReadersA(hContext, vbNullString, strReaders, readerBufferLnth)
    Catch err As Exception
    End Try
    Readers = strReaders.Split(Chr(0))
    strReaders = ""
    crdReaders = New Collection()
    For Each strReaders In Readers
        If strReaders.Trim.Length <> 0 Then crdReaders.Add(strReaders)
    Next
End Sub
```

```
*****
' The following procedures functions were created by Barry Hostead
*****
```

```
Private Sub initialiseReaders()
    For iCount = 1 To getCardReaders().Count
        appendLog()
        fileWriter.WriteLine(getCardReaders.Item(iCount))
        closeLog()
    Next iCount
End Sub

Public Sub connectToCard()
    Dim defaultReader As String
    Dim count As Integer
    count = getCardReaders.Count
    defaultReader = getCardReaders.Item(count)
    Dim Protocol As Integer
    Protocol = (GetProtocol)
    appendLog()
    ReturnCode = winSCard.SCardConnectA(hContext, defaultReader, SCARD_SHARE_EXCLUSIVE,
    SCARD_PROTOCOL_T0 Or SCARD_PROTOCOL_T1, hCard, Protocol)
```

```

If ReturnCode <> SCARD_S_SUCCESS Then
    Dim cError As New clsErrorMessage(ReturnCode)
    Throw New Exception(cError.errMessage)
    fileWriter.WriteLine("Error Connecting to reader" & cError.errMessage)
Else
    MessageBox.Show(hCard)
    fileWriter.WriteLine("Connection OK")
End If
closeLog()
End Sub

Private Sub beginTransaction()
    'Dim hCard As Long
    appendLog()
    ReturnCode = winSCard.SCardBeginTransaction(hCard)
    If ReturnCode <> SCARD_S_SUCCESS Then
        Dim cError As New clsErrorMessage(ReturnCode)
        Throw New Exception(cError.errMessage)
    Else
        fileWriter.WriteLine("SCardBeginTransaction OK")
    End If
    closeLog()
End Sub

Private Sub getStatus()
    Dim ATR(33) As Byte
    Dim ATRLen As Long
    Dim State As Long
    Dim Protocol As Long
    Dim iCount As Integer
    Dim strTempStatus As String

    appendLog()
    ATRLen = SCARD_ATR_LENGTH ' Must place length of the ATR buffer
    ' in order for the ATR return the right ATR string
    ReturnCode = winSCard.SCardState(hCard, State, Protocol, ATR(1), ATRLen)
    If ReturnCode <> SCARD_S_SUCCESS Then
        Dim cError As New clsErrorMessage(ReturnCode)
        Throw New Exception(cError.errMessage)
    Else
        fileWriter.WriteLine("SCardState OK")
        strTempStatus = ""
        For iCount = 1 To ATRLen
            strTempStatus = strTempStatus + Format(Hex(ATR(iCount)), "") + " "
        Next iCount
        fileWriter.WriteLine("ATR : " + strTempStatus)
        fileWriter.WriteLine("ATR Length : " + Str(ATRLen))
    End If
    closeLog()
End Sub

```

```
Private Sub TransmitAPDU(ByRef apduCommand As APDUREC)
    Dim SendRequest As SCARD_IO_REQUEST
    Dim RecvRequest As SCARD_IO_REQUEST
    Dim SendBuff(255 + 5) As Byte
    Dim SendBuffLen As Integer
    Dim RecvBuff(255 + 2) As Byte
    Dim RecvBuffLen As Integer
    Dim sTemp As String

    SendBuff(0) = apduCommand.bCLA
    SendBuff(1) = apduCommand.bINS
    SendBuff(2) = apduCommand.bP1
    SendBuff(3) = apduCommand.bP2
    SendBuff(4) = apduCommand.bP3

    If apduCommand.IsSend Then
        For iCount = 1 To apduCommand.bP3
            SendBuff(5 + iCount) = apduCommand.Data(iCount)
        Next iCount
        SendBuffLen = 5 + apduCommand.bP3
        RecvBuffLen = 2
    Else
        SendBuffLen = 5
        RecvBuffLen = 2 + apduCommand.bP3
    End If

    SendRequest.dwProtocol = SCARD_PROTOCOL_T0
    SendRequest.cbPciLength = Len(SendRequest)

    RecvRequest.dwProtocol = SCARD_PROTOCOL_T0
    RecvRequest.cbPciLength = Len(RecvRequest)

    ReturnCode = winSCard.SCardTransmit _
        ( _
            hCard, _
            SendRequest, _
            SendBuff(0), _
            SendBuffLen, _
            RecvRequest, _
            RecvBuff(0), _
            RecvBuffLen _
        )
    If ReturnCode <> SCARD_S_SUCCESS Then
        Dim cError As New clsErrorMessage(ReturnCode)
        Throw New Exception(cError.errMessage)
    Exit Sub
    Else
        fileWriter.WriteLine("SCardTransmit OK")
        closeLog()
    End If
```

```

sTemp = ""
For iCount = 1 To SendBuffLen

    sTemp = sTemp + Format(Hex(SendBuff(iCount)), "") + " "
Next
fileWriter.WriteLine("Send Buffer : " + sTemp)

sTemp = ""
For iCount = 1 To RecvBuffLen
    sTemp = sTemp + Format(Hex(RecvBuff(iCount)), "") + " "
Next
fileWriter.WriteLine("Receive Buffer : " + sTemp)

If Not apduCommand.IsSend Then
    For iCount = 1 To apduCommand.bP3 + 2
        apduCommand.Data(iCount) = RecvBuff(iCount)
    Next iCount
End If
closeLog()
End Sub

Public Sub getUserID()
    Dim apduCommand As APDURec
    ReDim apduCommand.Data(300)
    Dim iCount As Integer
    Dim sTemp As String
    'Select File
    Try
        apduCommand.bCLA = &H80
        apduCommand.bINS = &HA4
        apduCommand.bP1 = &H0
        apduCommand.bP2 = &H0
        apduCommand.bP3 = &H2
        'file to select
        apduCommand.Data(0) = &HFF
        apduCommand.Data(1) = &H0
        apduCommand.IsSend = True
        TransmitAPDU(apduCommand)
    Catch
    End Try
    'If cardResponse = "90 00" Then
    ' 'Read ID From File
    ' apduCommand.bCLA = &H80
    ' apduCommand.bINS = &HB2
    ' apduCommand.bP1 = &H0
    ' apduCommand.bP2 = &H0
    ' apduCommand.bP3 = &H8
    ' apduCommand.IsSend = False

    ' TransmitAPDU(apduCommand)
    ' cardResponse = userID

```

```
' MessageBox.Show(userID)
' End If
End Sub

Public Sub establishSession()
    establishContext()
    initialiseReaders()
    connectToCard()
End Sub

Public Sub startTransaction()
    beginTransaction()
    getStatus()
    getUserID()
End Sub

Public Sub endSession()
    ReturnCode = winSCard.SCardEndTransaction(hCard, SCARD_LEAVE_CARD)
    ReturnCode = winSCard.SCardEndTransaction(hCard, SCARD_LEAVE_CARD)
    ReturnCode = winSCard.SCardDisconnect(hCard, SCARD_UNPOWER_CARD)
End Sub
#End Region

End Class
```