

BasicCard ActiveX Control

Visual Basic Programming Manual

Version: 4.30

Date: 05.03.02 19:10

Author: Michael Petig, Robert Kazimirski

email: development@zeitcontrol.de

Web sites:

<http://www.zeitcontrol.de>

<http://www.basiccard.com>

<http://www.basiccardusa.com>

Contents

BasicCard ActiveX Control.....	1
Visual Basic Programming Manual.....	1
1. Overview	1
1.1 Design overview	1
1.1.1 BasicCard ActiveX Interface ZCBCardX.....	1
2. Developers Guide	2
2.1 Setup of Build Environment for Visual Basic 6.0	2
2.2 Setup of Runtime Environment	2
2.2.1 ZeitControl libraries:	2
2.2.2 Microsoft libraries:	2
2.3 Redistribute Runtime Environment	2
2.4 Programming Intro	3
2.4.1 Initialization Code	3
2.4.2 Examples in Visual Basic	3
3. Reference.....	6
3.1 Class zcBasicCard	6
3.1.1 Class Members by Category	6
3.1.1.1 Reader Management	6
3.1.1.2 Name Services	6
3.1.1.3 Reader Access Services	6
3.1.1.4 Connection Services	6
3.1.1.5 Low Level Transaction Functions	6
3.1.1.6 Debug Support Functions	6
3.1.1.7 Attachment Functions.....	6
3.1.1.8 Key Management Functions	6
3.1.1.9 Transaction Functions.....	6
3.1.1.10 Eeprom Access Functions.....	7
3.1.1.11 Encryption Functions.....	7
3.1.1.12 Card Management Functions.....	7
3.1.1.13 File System Functions.....	7
3.1.1.14 Error Properties.....	8
3.1.2 Reader Functions by Name.....	8
3.1.2.1 CloseReader.....	8
3.1.2.2 Connect.....	8
3.1.2.3 DeregisterService.....	9
3.1.2.4 Disconnect	10
3.1.2.5 DefaultReader.....	10
3.1.2.6 FastInit	11
3.1.2.7 OpenReader	11
3.1.2.8 ReaderCount	12
3.1.2.9 ReaderName	12
3.1.2.10 Reconnect	13
3.1.2.11 RegisterService.....	13
3.1.2.12 SelectReaderDialog	14
3.1.2.13 LogFile.....	14
3.1.2.14 ExchangeAPDU.....	15
3.1.2.15 WaitCard.....	16
3.1.3 Card Functions by Name	17
3.1.3.1 ApplicationID	17
3.1.3.2 Attach.....	17
3.1.3.3 Attribute.....	18
3.1.3.4 ClearEeprom.....	18
3.1.3.5 CloseAllFiles	19
3.1.3.6 CloseFile.....	20
3.1.3.7 CurDir.....	20
3.1.3.8 CustomLock.....	21
3.1.3.9 Detach.....	22
3.1.3.10 DirCount	22
3.1.3.11 DirFile.....	23
3.1.3.12 Echo.....	24

3.1.3.13	EepromCRC.....	24
3.1.3.14	EepromSize.....	25
3.1.3.15	EepromStart.....	26
3.1.3.16	EndEncryption.....	26
3.1.3.17	EraseFile.....	27
3.1.3.18	FileErr.....	27
3.1.3.19	FileGet.....	27
3.1.3.20	FileGetPos.....	29
3.1.3.21	FileLength.....	29
3.1.3.22	FileLock.....	30
3.1.3.23	FileLock2.....	32
3.1.3.24	FilePrint.....	33
3.1.3.25	FilePut.....	33
3.1.3.26	FilePutPos.....	34
3.1.3.27	FileRead.....	35
3.1.3.28	FileReadLine.....	35
3.1.3.29	FileReadLong.....	36
3.1.3.30	FileReadSingle.....	37
3.1.3.31	FileReadString.....	37
3.1.3.32	FileReadUser.....	38
3.1.3.33	FileReadUser2.....	38
3.1.3.34	FileWrite.....	39
3.1.3.35	FileWriteLong.....	39
3.1.3.36	FileWriteSingle.....	40
3.1.3.37	FileWriteString.....	40
3.1.3.38	FileWriteUser.....	41
3.1.3.39	FilePos.....	41
3.1.3.40	LastErr.....	42
3.1.3.41	MkDir.....	42
3.1.3.42	OpenFile.....	42
3.1.3.43	OpenFile2.....	44
3.1.3.44	QueryEof.....	45
3.1.3.45	Param1, Param2, ...,Param12.....	46
3.1.3.46	ReadEeprom.....	48
3.1.3.47	ReadKeyFile.....	48
3.1.3.48	ReadLock.....	49
3.1.3.49	Rename.....	49
3.1.3.50	Rmdir.....	50
3.1.3.51	SetKey.....	50
3.1.3.52	SetPoly.....	51
3.1.3.53	StartEncryption.....	51
3.1.3.54	State.....	52
3.1.3.55	SW1SW2.....	53
3.1.3.56	Transaction.....	53
3.1.3.57	Transaction2.....	56
3.1.3.58	WriteEeprom.....	57
3.1.3.59	WriteLock.....	57
3.1.3.60	Version.....	58
3.1.4	Constants.....	59
3.1.4.1	BASICCARDSTATE.....	59
3.1.4.2	BASICCARDLC.....	59
3.1.4.3	BASICCARDLE.....	59
3.1.4.4	BASICCARDERROR.....	59
3.1.4.5	BASICCARDFILEMODE.....	61
3.1.4.6	BASICCARDFILEACCESS.....	61
3.1.4.7	BASICCARDFILESHARE.....	61
3.1.4.8	BASICCARDSW1SW2.....	62
3.1.4.9	BASICCARDFILEERROR.....	62
3.1.4.10	BASICCARDFILEATTRIB.....	62
3.1.4.11	BASICCARDFILELOCKINFO.....	62
3.1.4.12	BASICCARDFILECUSTOMLOCKINFO.....	63
3.1.4.13	BASICCARDENCRYPTION.....	63

1. Overview

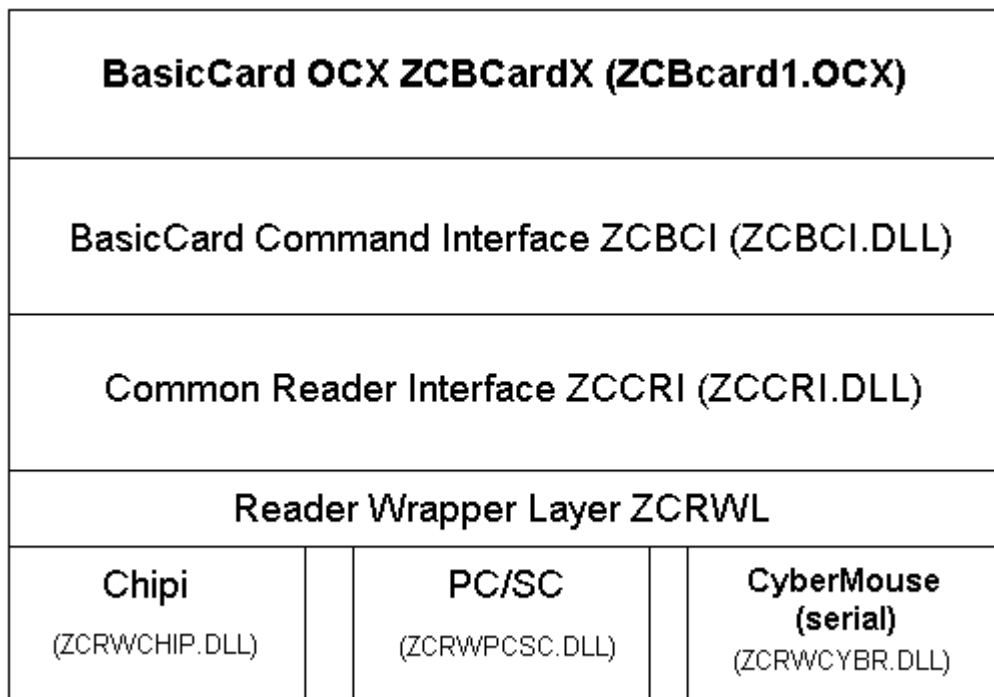
This document describes the BasicCard ActiveX control. This ActiveX is enabled in Visual Basic™ and many other environments. This ActiveX control enables developers to use BasicCard from inside their Visual Basic programs BasicCard PC applications almost as simply as if using ZCBasic (ZeitControl Basic). This ActiveX control is currently available for Windows™ 32 bit operating systems. The control supports the ZeitControl CyberMouse card reader, ZeitControl Chipi card reader and PC/SC compatible readers on these operating systems. It is possible to plug in different readers by creating a small support library (must be developed using C/C++).

This ActiveX control uses our C-API. For reference how to use the C-API please refer to documentation include for this purpose.

Note: To understand this document it is strongly recommended to read the BasicCard manual first since it is the elementary manual for BasicCard operating system. Also if you are familiar with C language it would be useful (but not necessary) to read C-API manual.

1.1 Design overview

This ActiveX consists of one file named ZCBCard1.OCX, which is placed as a fourth layer of C-API layers.



Picture 1

1.1.1 BasicCard ActiveX Interface ZCBCardX

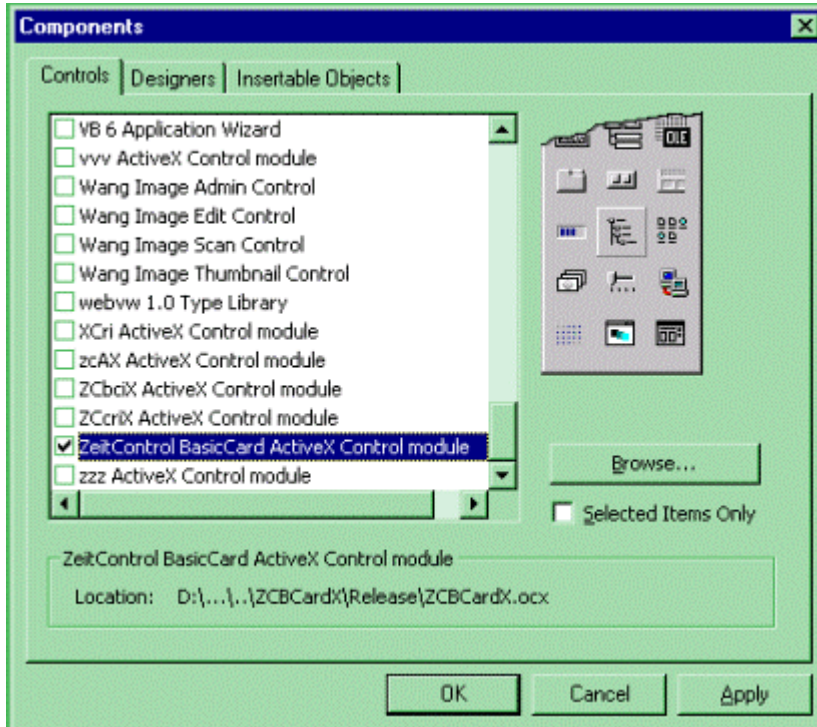
ZCBCardX consists of one class zcBasicCard which includes both reader and card interface. Almost all predefined commands of Enhanced and Compact BasicCard (including full file system support for Enhanced BasicCard) are supported when using this class. Further a simple transaction command enables you to create access function to user defined card commands. As in ZCBasic transparent encryption is included in zcBasicCard. To enable this, additional functions to load and set keys and polynomials are available. To connect to the reader device, functions to attach to and detach from the device are provided.

2. Developers Guide

2.1 Setup of Build Environment for Visual Basic 6.0

To install ZCBCard1.OCX you need to setup the API/OCX development package. In next step you should add ZCBCardX to your Visual Basic project. To do this select Project/Components... from the menu.

Note: If you have installed the beta version of the OCX before, be careful to select "ZeitControl BasicCard Active Control module **V1.0**".



2.2 Setup of Runtime Environment

After finishing the installation of the BasicCard ActiveX control the required files are copied to your system directory:

2.2.1 ZeitControl libraries:

- ZCBCARD1.OCX
- ZCBCI.DLL
- ZCCRI.DLL
- ZCRWSER.DLL
- ZCRWPCSC.DLL requires PC/SC support to be installed

2.2.2 Microsoft libraries:

- OLEPRO32.DLL, ..

2.3 Redistribute Runtime Environment

We have decided to make it as easy as possible for you to include necessary files in your own setup programs. For this purpose we have decided to support new "Windows Installer" API. Take a look at http://www.installsite.org/w2k_msiauth.htm for products supporting this API. Following files (merge modules) may be used to be included in your own setup routines:

ZCBCARD1.MSM

ZCBCI.MSM

ZCCRI.MSM

ZCRWSER.MSM (required when using ZeitControl CyberMouse or Chipi serial card reader without PC/SC)

ZCRWPCSC.MSM (required when using other PC/SC card readers)

ZCPLIB.DLL (general runtime DLL used by other DLLs)

Additionally the following Microsoft module is required:

OLEAUT32.MSM

There is a separate runtime setup available on CD and at <http://www.basiccard.com>.

2.4 Programming Intro

2.4.1 Initialization Code

To use OCX you must first create an object from `zcBasicCard` class. The following example shows a way to do it:

```
Dim zc As New zcBasicCard
```

2.4.2 Examples in Visual Basic

Before you can call any BasicCard command you must first open a reader device and establish a connection to the card in reader. Using `zcBasicCard` class is easier. By default you don't need to worry how, where and when to open a reader, connect to a card in a reader or attach the card to the reader – it is done automatically for you. Anyway it is recommended to open and attach the reader manually. When you want to decrease the initialization time of BasicCard OCX you should also use "FastInit" which disables scanning of ports for connected readers.

```
Dim zc As zcBasicCard           ' declare a variable

Private Sub Form_Load()
    Dim Error as Long
    Call zc.FastInit ' Must be first call to control
    Error=zc.OpenReader("") ' Open default reader
    if Error<>0 then
        ' Do error handling here
    end if
    Error=zc.Attach ' Attach to API
    if Error<>0 then
        ' Do error handling here
    end if
End Sub

Private Sub Form_Unload(Cancel as Integer)
    Call zc.CloseReader ' close card reader
End Sub

Private Sub Command1_Click()
    Print zc.Echo(1, "abc") 'call standard echo command with increment 1
                          'result "bcd"
End Sub
```

Another example shows a way to call user defined commands. It is recommended to encapsulate this command as a separate function.

For example if there is a card command which is defined in ZCBasic as follows

```
declare Command &H20 &H14 UserCommand(FixStr as string*25, VarLong&,_
                                       VarInt%, VarByte@, VarStr$)
```

your function in Visual Basic may look like this:

2. Developers Guide

```
Dim zc As New zcBasicCard

Private Sub Form_Load()
    Dim Error as Long
    Call zc.FastInit ' Must be first call to control
    Error=zc.OpenReader("") ' Open default reader
    if Error<>0 then
        ' Do error handling here
    end if
    Error=zc.Attach ' Attach to API
    if Error<>0 then
        ' Do error handling here
    end if
End Sub

Private Sub Form_Unload(Cancel as Integer)
    Call zc.CloseReader ' close card reader
End Sub

Function UserCommand(FixStr As String, VarLong As Long, VarInt As Integer,
VarByte As Byte, VarStr As String)
' Size of FixStr must be according to BasicCard user command declare
If Len(FixStr) <> 25 Then
    'do error handling here
End If
' First you need to assign a value to Param1,...
zc.Param1 = FixStr
zc.Param2 = VarLong
zc.Param3 = VarInt
zc.Param4 = VarByte
zc.Param5 = VarStr
'Call transaction with Cla = &H20 and Ins = &H14, give parameter count 5
Call zc.Transaction2(&H20, &H14, 5)
'Check return error
If zc.LastErr <> 0 Then
    'Do error handling here
    exit function
End If
'if it's desired copy returned parameter back to function parameters
FixStr = zc.Param1
VarLong = zc.Param2
VarInt = zc.Param3
VarByte = zc.Param4
VarStr = zc.Param5
End Function
```

Note: It's necessary to assign a value to Param1,... before calling Transaction, Transaction2. Otherwise you get error &H2000000 (ERROR_PARM). This is because Transaction, Transaction2 needs to know the type of the parameters Param1, ...

There are two choices to pass parameter data declared as a string (like "*FixStr As String*5*" or "*VarStr As String*") to the OCX. You can either use a normal Visual Basic string or you can use a one dimensional array of bytes (like "*ReDim MyBytes(1 To 5) as Byte*"). When using strings you should be aware that Visual Basic uses 2 bytes for every character by using Unicode characters. Because strings are often used to pass binary data from and to the BasicCard no translation from Unicode to Multibyte characters is done. So BasicCard OCX does only accept strings including characters in range 0 to 255. If strings outside this value are passed BASICCARDERROR.BCI_ERROR_PARM is returned. When using Visual Basic version 6.0 or above we strongly recommend to use an array of bytes instead of strings to pass binary data to BasicCard OCX. When using Visual Basic version 5.0 we recommend to use strings, at least in conjunction with ParamX properties. This is because when assigning an array of bytes to a parameter result returned will also be array of bytes. This

seems to cause problems in Visual Basic 5.0 which will fail to assign return values like this to byte array variables.

3. Reference

3.1 Class *zcBasicCard*

3.1.1 Class Members by Category

3.1.1.1 Reader Management

DefaultReader

RegisterService

DeregisterService

3.1.1.2 Name Services

ReaderCount

ReaderName

SelectReaderDialog

3.1.1.3 Reader Access Services

FastInit

OpenReader

CloseReader

3.1.1.4 Connection Services

WaitCard

Connect

Reconnect

Disconnect

3.1.1.5 Low Level Transaction Functions

ExchangeAPDU

3.1.1.6 Debug Support Functions

LogFile

3.1.1.7 Attachment Functions

Attach

Detach

3.1.1.8 Key Management Functions

ReadKeyFile

SetKey

SetPoly

3.1.1.9 Transaction Functions

Transaction

Transaction2

3.1.1.10 Eeprom Access Functions

EepromCRC

EepromSize

ClearEeprom

ReadEeprom

WriteEeprom

3.1.1.11 Encryption Functions

StartEncryption

EndEncryption

3.1.1.12 Card Management Functions

ApplicationID

Echo

State

3.1.1.13 File System Functions

CloseAllFiles

CloseFile

CurDir

DirCount

DirFile

EraseFile

FileGet

FileGetPos

FileLength

FilePrint

FilePut

FilePutPos

FileRead

FileReadLine

FileReadSingle

FileReadString

FileReadUser

FileReadUser2

FileWrite

FileWriteLong

FileWriteSingle

FileWriteString

FileWriteUser

3. Reference

Attribute

FilePos

MkDir

OpenFile

QueryEof

Rename

Rmdir

3.1.1.14 Error Properties

LastErr

SW1SW2

FileErr

3.1.2 Reader Functions by Name

3.1.2.1 CloseReader

Close reader device.

Syntax: (Method)

Object.CloseReader

Parameter: none

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful

See also: OpenReader

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
....
Error=zc.OpenReader
if Error<>0 then
    ' Do error handling here
end if
```

3.1.2.2 Connect

Establish connection to card, return ATR.

Syntax: (Method)

Object.Connect as String

Parameter: none

Return value:

Card ATR. Binary data (array of bytes) using Visual Basic Variant type.

Error values available by LastErr: (see LastErr)

Value	Description
-------	-------------

BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.CRI_ERROR_UNKNOWN_READER	Reader not known
BASICCARDERROR.CRI_ERROR_READER_BUSY	Reader busy
BASICCARDERROR.CRI_ERROR_NO_CARD	No card in reader

See also: Reconnect, Disconnect

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim ATR as Variant
Dim ATRDump as String
Dim i as Long
....
ATR=zc.Connect
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if
ATRDump=""
for i=LBound(ATR) to UBound(ATR)
    ATRDump=ATRDump+Hex$(ATR(i))+ " "
next i
```

3.1.2.3 DeregisterService

Deregister RWL service DLL.

Syntax: (Method)

Object.DeregisterService (Service as String)

Parameter:

Number	Name	Direction	Description
1	Service	in	Name of service to deregister

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.CRI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.CRI_ERROR_MISC	Unknown system error

See also:

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
....
Error=zc.DeregisterService("PCSC") ` deregister PCSC service DLL
if Error<>0 then
    ` Do error handling here
end if
```

3. Reference

3.1.2.4 Disconnect

Disconnect from card in reader device.

Syntax: (Method)

Object.Disconnect

Parameter: none

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful

See also: Connect, Reconnect

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
....
Error=zc.Disconnect
if Error<>0 then
    ` Do error handling here
end if
```

3.1.2.5 DefaultReader

Get/Set default reader. Default reader is stored inside Windows™ registry. It is used when empty String is passed as reader name when calling OpenReader. Or if reader is opened automatically because OpenReader is not called.

Syntax: (Property, Read/Write)

Object.DefaultReader as String [=NewDefReader]

Content:

Name of default reader.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, property call successful
BASICCARDERROR.CRI_ERROR_PARM	Invalid value passed to property call
BASICCARDERROR.CRI_ERROR_NO_DEFAULT	No default reader defined
BASICCARDERROR.CRI_ERROR_MISC	Unknown system error

See also: ReaderCount

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
....
zc.DefaultReader="CyberMouse:COM1"
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if
```

3.1.2.6 FastInit

Notify card reader interface to make fast initialization by not scanning for connected readers. We recommend to use this function as the first call when there is no need to show or query connected readers, for example because the default reader is used. Without this call, first call to OCX which starts initialization, may take a long time. This time may also be decreased by unregistering unused reader service DLLs.

Syntax: (Method)

Object.FastInit

Parameter: none

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful

See also:

Example: see OpenReader

3.1.2.7 OpenReader

Open reader device

Syntax: (Method)

Object.OpenReader (Name as String)

Parameter:

Number	Name	Direction	Description
1	Name	in	Name of reader device to open. ""(empty) for default reader.

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.CRI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.CRI_ERROR_UNKOWN_READER	Reader not known or not supported
BASICCARDERROR.CRI_ERROR_READER_BUSY	Reader busy or not accessible

See also: CloseReader, ReaderName, SelectReaderDialog

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
....
Private Sub Form_Load()
    Error=zc.FastInit ` Do fast initialization
    if Error<>0 then
        ` Do error handling here
    end if
    Error=zc.OpenReader("") ` Open default reader
    if Error<>0 then
        ` Do error handling here
    end if
end Sub
```

3. Reference

```
end if
Error=zc.Attach() ` Attach to API
if Error<>0 then
    ` Do error handling here
end if
End Sub

Private Sub Form_Unload(Cancel As Integer)
    Call zc.CloseReader()
End Sub
```

3.1.2.8 ReaderCount

Get number of supported reader devices.

Syntax: (Property, Read only)

Object.ReaderCount as Long

Content:

Number of supported reader devices

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful

See also: ReaderName, SelectReaderDialog

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim RCount as Long
Dim ReaderList as String
Dim NextReader as String
Dim i as Long
....
RCount=zc.ReaderCount
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if
ReaderList=""
for i=0 to RCount-1
    NextReader= zc.ReaderName(i)
    Error=zc.LastErr
    if Error<>0 then
        ` Do error handling here
    end if
    ReaderList=ReaderList+ NextReader + " "
next i
```

3.1.2.9 ReaderName

Get name of supported reader addressed by number *Num*.

Syntax: (Property, Read only)

Object.ReaderName (Num as Long) as String

Content:

Name of requested reader

Parameter:

Number	Name	Direction	Description
--------	------	-----------	-------------

1	Num	In	Number of requested reader (0 .. readercount-1)
---	-----	----	---

Error values: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.CRI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.CRI_ERROR_UNKOWN_READER	Reader not known (Num >= readercount)

See also: SelectReaderDialog

3.1.2.10 Reconnect

Reconnect to card in reader or in other words - reset a card.

Syntax: (Method)

Object.Reconnect as String

Return values:

Card ATR. Binary data.

Error values: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.CRI_ERROR_NO_CARD	No card in reader

See also: Connect, Disconnect

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim ATR as Variant
Dim ATRDump as String
Dim i as Long
....
ATR=zc.Reconnect
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if
ATRDump=""
for i=LBound(ATR) to UBound(ATR)
    ATRDump=ATRDump+Hex$(ATR(i))+ " "
next i
```

3.1.2.11 RegisterService

Register a RWL service DLL.

Syntax: (Method)

Object.RegisterService (Service as String, DLLName as String)

Parameter:

Number	Name	Direction	Description
1	Service	In	Name of service to register (must be the same as used in

3. Reference

			RWL DLL).
2	DLLName	In	Name of DLL implements this service

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.CRI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.CRI_ERROR_MISC	Unknown system error

See also: DeregisterService

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
....
REM register PCSC service DLL
Error=zc.RegisterService("PCSC", "ZCRWPCSC.DLL")
if Error<>0 then
    ` Do error handling here
end if
```

3.1.2.12 SelectReaderDialog

Show dialog to select a reader device.

Syntax: (Property, Read only)

Object.SelectReaderDialog as String

Content:

Name of selected reader device to open.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.CRI_CANCEL	Dialog cancelled, no reader selected

See also: ReaderName

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim ReaderName as String
....
ReaderName=zc.SelectReaderDialog
Error=zc.LastErr
if Error=BASICCARDERROR.CRI_NOERROR then
    Error=zc.OpenReader(ReaderName) ` Open selected reader
end if
```

3.1.2.13 LogFile

Log all input and output data to log file.

Syntax: (Property, Write only)

Object.LogFile as String [= LogFile]

Content:

Name of log file, ""(empty) to disable logging

Error values: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.CRI_ERROR_MISC	Unable to open file

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
....
zc.LogFile="c:\io.log" \ open log file
Error=zc.LastErr
if Error<>0 then
    \ Do error handling here
end if

zc.LogFile="" \ close log file
```

3.1.2.14 ExchangeAPDU

Send (and receive) T1 transaction to (from) smart card.

Note: This function is for low level use as by ZCBCI-API. Use Transaction, Transaction2 instead of this function, whenever possible.

Syntax: (Method)

Object.ExchangeAPDU (DataIn as Variant) as Variant

Parameter:

Number	Name	Direction	Description
1	DataIn	In	Binary data to sent to smart card either as string or as array of bytes.

Return values:

Binary data received from smart card. If input data (DataIn) is of type String return data also is of type string otherwise return data is returned as array of bytes.

Error values: (see LastErr)

BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.CRI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.CRI_ERROR_PROTOCOL	Protocol (T1) error transaction failed
BASICCARDERROR.CRI_ERROR_NO_CARD	No card in reader, card removed
BASICCARDERROR.CRI_ERROR_OVERFLOW	Return value overflow

See also: Transaction, Transaction2

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim APDU as Variant
```

3. Reference

```
Dim Reply as Variant
Dim Data(1 To 8) as Byte
....
REM Echo command
Data(1)=&HC0 ` CLA
Data(2)=&H14 ` INS
Data(3)=1 ` P1
Data(4)=0 ` P2
Data(5)=3 ` Lc
Data(6)=1 ` IData
Data(7)=2
Data(8)=3
Data(9)=3 ` Le
APDU = Data;
Reply=zc.ExchangeAPDU(APDU) ` open log file
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if

zc.LogFile="" ` close log file
```

3.1.2.15 WaitCard

Wait for card to be inserted. This function will wait a maximum of Timeout ms before returning. When a card is inserted or timeout is exceeded function returns. Calling this function will block Windows message queue as long as function is active. So this function should not be called with timeouts > 30 ms. Instead of this a timer should be created and this function should be called inside time function using a timeout of 0. Then you can take any action depending on whether card inserted or removed.

Syntax: (Method)

Object.WaitCard (Timeout as Long)

Parameter:

Number	Name	Direction	Description
1	Timeout	In	Maximum time to wait (ms). Use -1 for indefinite wait.

Return value: error code (see below)

Error values: (see LastErr)

BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.CRI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.CRI_ERROR_NO_CARD	No card in reader, card removed
BASICCARDERROR.CRI_ERROR_READER_BUSY	Card reader busy

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim CardInReader as Boolean
....
Private Sub Form_Load
    ...
    CardInReader=False;
    Timer1.Enabled=True;
End Sub

Private Sub Timer1_Timer()
```

```

Dim Ret as Long
Ret=zc.WaitCard(0)
if Ret=BASICCARDERROR.CRI_NOERROR then
    REM Card is inserted
    if CardInReader=False then
        Call ReadCard(...)
    end if
    CardInReader=True
else
    CardInReader=False
end if
End Sub

```

3.1.3 Card Functions by Name

3.1.3.1 ApplicationID

BasicCard GET APPLICATION ID command (Cla C0, Ins 0E). Get application id.

Syntax: (Property, Read only)

Object.ApplicationID as String

Content:

Application ID.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.

See also:

Supported cards: ZC1.X, ZC2.X, ZC3.X

Example:

```

Dim zc as New zcBasicCard
Dim Error as Long
Dim AppID as String
....
AppID=zc.ApplicationID
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if

```

3.1.3.2 Attach

Attach reader device to ZCBCI card handle. Must be first call before you call any other function. It's only for compatibility to C-API. Usually you never need to call this function.

Syntax: (Method)

Object.Attach

Return value: error code (see below)

Error values returned:

3. Reference

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful

See also: Detach

Supported cards: ZC1.X, ZC2.X, ZC3.X

Example: see OpenReader

3.1.3.3 Attribute

Get file or directory attributes.

Syntax: (Property, Read only)

Object.Attribute(Path as String) as Long

Content:

Returned attributes (either BASICCARDFILEATTRIB.CardDirectory or BASICCARDFILEATTRIB.CardFile)

Parameter:

Number	Name	Direction	Description
1	Path	In	File or directory to query

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also:

Supported cards: ZC2.X, ZC3.X

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim Attrib as Long
....
Attrib=zc.Attribute("\Test")
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if
if BASICCARDFILEATTRIB.CardDirectory=Attrib then
    ` \Test it is a directory on card
else
    if BASICCARDFILEATTRIB.CardFile=Attrib then
        ` \Test it is a file on card
    end if
end if
```

3.1.3.4 ClearEeprom

BasicCard CLEAR EEPROM command (Cla C0, Ins 04). Clear (set to FF) length bytes of eeprom starting at address start.

Note: This command is required when loading an application (image) into a card. There is no use for this function for normal applications.

Note: Keep in mind that CLEAR EEPROM will not send WTX, so do not clear more than 1 kB of eeprom memory with one call.

Syntax: (Method)

Object.ClearEeprom (Start as Long, Length as Long)

Parameter:

Number	Name	Direction	Description
1	Start	In	Start address of eeprom memory to clear
2	Length	In	Number of bytes to clear

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.

See also: EepromSize, EepromCRC, WriteEeprom, ReadEeprom

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.1.3.5 CloseAllFiles

Close all open card files on BasicCard.

Syntax: (Method)

Object.CloseAllFiles

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: OpenFile, OpenFile2, CloseFile

Supported cards: ZC2.X, ZC3.X

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
....
Error=zc.CloseAllFiles
if Error<>0 then
    ' Do error handling here
```

3. Reference

end if

3.1.3.6 CloseFile

Close open card file on BasicCard.

Syntax: (Method)

Object.CloseFile (FileNum as Long)

Parameter:

Number	Name	Direction	Description
1	FileNum	In	File number of file to close.

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: CloseAllFiles, OpenFile, OpenFile2

Supported cards: ZC2.X, ZC3.X

Example: see OpenFile2

3.1.3.7 CurDir

Query/Change current directory on card.

Syntax: (Property, Read/Write)

Object.CurDir (Path as String) as String [=NewCurrDir]

Content:

Current directory on card.

Parameter:

Number	Name	Direction	Description
1	Path	In	New path to current directory on card.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)
BASICCARDERROR.BCI_ERROR_OVERFLOW	Directory name exceeds buffer size

See also: DirCount, DirFile

Supported cards: ZC2.X, ZC3.X

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim CurDir as String
....
CurDir=zc.CurDir ` Get current directory
Error=zc.LastErr
if Error<>0 then
  ` Do error handling here
end if
....
zc.CurDir="\Test" ` Set current directory
Error=zc.LastErr
if Error<>0 then
  ` Do error handling here
end if
```

3.1.3.8 CustomLock

Get/Set Custom lock information of file or directory.

Syntax: (Property, Read/Write)

Object: **CustomLock**(Path as String) as Long

Content:

CustomLock (either BASICCARDFILECUSTOMLOCKINFO.CLocked or BASICCARDFILECUSTOMLOCKINFO.CUnlocked)

Parameter:

Number	Name	Direction	Description
1	Path	In	File or directory to set
2	CustomLock		either BASICCARDFILECUSTOMLOCKINFO.CLocked or BASICCARDFILECUSTOMLOCKINFO.CUnlocked see constants

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: FileLock, FileLock2, ReadLock, WriteLock

Supported cards: ZC2.X, ZC3.X

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim LockInfo as Long
....
LockInfo=zc.CustomLock("\Test") ` get current lock information
```

3. Reference

```
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if
....
REM Lock file \Test, must be unlocked from inside card
zc.CustomLock( "\Test" )=BASICCARDFILECUSTOMLOCKINFO.CLocked
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if
```

3.1.3.9 Detach

Detach BasicCard from reader device. It's only for compatibility to C-API. Usually you never need to call this function.

Syntax: (Method)

Object.Detach

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful

See also: Attach

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.1.3.10 DirCount

Count files and directories on card matching file specification.

Syntax: (Property, Read only)

Object.DirCount (FileSpec as String) As Long

Content:

Number of files or directories match given specification

Parameter:

Number	Name	Direction	Description
1	FileSpec	In	File specification to search for. For example "\Test*"

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: DirFile

Supported cards: ZC2.X, ZC3.X

Example:

```

Dim zc as New zcBasicCard
Dim Error as Long
Dim SW1SW2 as Long
Dim FileErr as Long
Dim Error as Long
Dim Count as Long
Dim FileDir as String
Dim i as Long
....
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
Count=zc.DirCount("\*") ` Count all files or directories in root
Error=zc.LastErr
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if
For i=1 to Count
    Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
    FileDir=zc.DirFile("\*", i) ` get file or directory (i) in root
    Error=zc.LastErr
    if Error<>0 then
        SW1SW2=zc.SW1SW2
        FileErr=zc.FileErr
        ` Do error handling here
    end if
    ....
next i

```

3.1.3.11 DirFile

Return file or directory name of card file or directory matching given file specification and number. DirCount must be called before with same file specification.

Syntax: (Property, Read Only)

Object.DirFile(FileSpec as String, Num as Long) as String

Content:

Name of matched directory or file

Parameter:

Number	Name	Direction	Description
1	FileSpec	In	File specification to search for. For example "\Test*"
2	Num	In	Number of files or directory to return

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)
BASICCARDERROR.BCI_ERROR_OVERFLOW	File name exceeds buffer size

See also: DirCount

Supported cards: ZC2.X, ZC3.X

3. Reference

Example: see DirCount

3.1.3.12 Echo

BasicCard ECHO command (Cla C0, Ins 04). Echo data (for testing).

Syntax: (Method)

Object.Echo (Inc as Integer, Test as String) as String

Parameter:

Number	Name	Direction	Description
1	Inc	In	Increment value to add to every byte in Test by BasicCard echo command
2	Test	In	String to Echo

Return values:

Echo string from card.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_OVERFLOW	Too much parameter data
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.
BASICCARDERROR.BCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also:

Supported cards: ZC1.X, ZC2.X, ZC3.X

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim MyEcho as String
....
MyEcho=zc.Echo(1,"ABCD") ` Echo with increment of one, A -> B, ...
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if
if "BCDE"<>MyEcho then
    ` Invalid echo returned
end if
```

3.1.3.13 EepromCRC

BasicCard EEPROM CRC command (Cla C0, Ins 0A). Calculate and return eeprom crc.

Note: This command is required when loading an application (image) into a card. There is no use for this function for normal applications.

Note: Calling EepromCRC on an empty (no program loaded) Enhanced BasicCard will destroy the card.

Note: Keep in mind EEPROM CRC will not send WTX, so a call of this command with a big length value may cause a timeout. There may be an enhanced version of EepromCRC, which will split this type of call to commands using less length values, in future.

Syntax: (Property, Read only)

Content:

Calculated CRC.

Object.EepromCRC (Start as Long, Length as Long) as Long

Parameter:

Number	Name	Direction	Description
3	Start	In	Eeprom address to start calculate
4	Length	In	Number of bytes to calculate CRC on

Return values:

Calculated CRC.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_OVERFLOW	Too much parameter data
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.
BASICCARDERROR.BCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also: EepromSize, EepromStart, ClearEeprom

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.1.3.14 EepromSize

BasicCard EEPROM SIZE command (Cla C0, Ins 02). ActiveX version returns only length of card eeprom. (see EepromStart)

Note: This command is required when loading an application (image) into a card. There is no use for this function for normal applications.

Syntax: (Property, Read only)

Object.EepromSize as Long

Content:

Length of BasicCard eeprom.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_OVERFLOW	Too much parameter data
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs

3. Reference

	from &H9000 or &H61XX.
BASICCARDERROR.BCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also: EepromStart, EepromCRC, ClearEeprom

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.1.3.15 EepromStart

BasicCard EEPROM SIZE command (Cla C0, Ins 02). ActiveX version returns only start of card eeprom.(See EepromSize)

Note: This command is required when loading an application (image) into a card. There is no use for this function for normal applications.

Syntax: (Property, Read only)

Object.EepromStart as Long

Content:

Eeprom start address.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_OVERFLOW	Too much parameter data
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.
BASICCARDERROR.BCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also: EepromSize, EepromCRC, ClearEeprom.

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.1.3.16 EndEncryption

BasicCard END ENCRYPTION command (Cla C0, Ins 12). Stop transparent encryption.

Syntax: (Method)

Object.EndEncryption

Error values: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.
BASICCARDERROR.BCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also: StartEncryption

Supported cards: ZC1.X, ZC2.X, ZC3.X

Example: see StartEncryption

3.1.3.17 EraseFile

Erase card file.

Syntax: (Method)

Object.EraseFile (Path as String)

Parameter:

Number	Name	Direction	Description
1	Path	In	File to remove

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: OpenFile, OpenFile2

Supported cards: ZC2.X, ZC3.X

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim SW1SW2 as Long
Dim FileErr as Long
Dim Error as Long
....
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
Error=zc.EraseFile("\Test") ` delete file \test
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if
```

3.1.3.18 FileErr

Return file error code which occurs when last function was called.

Syntax: (Property, Read only)

Object.**FileErr** as Long

Content:

File error code. One of BASICCARDFILEERROR.____.

See also: Constant, SW1SW2, LastErr

3.1.3.19 FileGet

Get content of open file at current file position.

3. Reference

Syntax: (Method)

Object.**FileGet** (FileNum as Long, Len as Integer) as Variant

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file.
2	Len	In	Number of bytes to read.

Return values:

Binary data read from file as byte array (LBound=1) stored in variant

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)
BASICCARDERROR.BCI_ERROR_MISSING_PARM	Not enough data received

See also: FileGetPos, FileRead, FileReadLine, FileReadLong, FileReadSingle, FileReadString, FileReadUser, FileReadUser2, OpenFile, OpenFile2

Supported cards: ZC2.X, ZC3.X

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim SW1SW2 as Long
Dim FileErr as Long
Dim Error as Long
Dim FileNum as Long
Dim Data as Variant
Dim DumpData as String
Dim LenOfFile as Long
Dim i as Long
....
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
REM Open file \Test in binary mode (put, get) for reading
FileNum=zc.OpenFile2("\Test", "FOR BINARY ACCESS READ")
Error=zc.LastErr
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
REM Get length of file
LenOfFile=zc.FileLength(FileNum)
Error=zc.LastErr
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
```

```

REM Read 10 bytes from file
Data=zc.FileGet(FileNum,10)
Error=zc.LastErr
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if
DumpData=""
For i=1 to 10
    DumpData=DumpData + Hex$(Data(i)) + " "
next i
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
REM Close file
Error=zc.CloseFile(FileNum)
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if

```

3.1.3.20 FileGetPos

Get content of open file at specified file position.

Syntax: (Method)

Object: **FileGetPos** (FileNum as Long, Pos as Long, Len As Integer) as Variant

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file
2	Pos	In	Requested read position
3	Len	In	Number of bytes to read.

Return values:

Binary data read from file as byte array (LBound=1) stored in variant

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)
BASICCARDERROR.BCI_ERROR_MISSING_PARM	Not enough data received

See also: FileGet, FileGetPos, FileRead, FileReadLine, FileReadLong, FileReadSingle, FileReadString, FileReadUser, FileReadUser2, OpenFile, OpenFile2

Supported cards: ZC2.X, ZC3.X

Example: see FileGet

3.1.3.21 FileLength

Return file length of open file.

3. Reference

Syntax: (Property, Read only)

Object: **FileLength** (FileNum as Long) as Long

Content:

Length of file (in byte).

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file to query

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: QueryEof

Supported cards: ZC2.X, ZC3.X

Example: see FileGet

3.1.3.22 FileLock

Get lock information of file or directory.

Syntax: (Property, Read)

Object: **FileLock**(Path as String) as Variant

Content:

Array of 7 (1 to 7) bytes. With following content:

Byte 1: Read lock condition:

- BASICCARDFILELOCKINFO.Allowed
Read access is allowed
- BASICCARDFILELOCKINFO.Keyed1
Read access is forbidden with exception: when encryption with key ReadKey1 is active, read access is allowed.
- BASICCARDFILELOCKINFO.Keyed2
Read access is forbidden with exception: when encryption with key ReadKey1 or ReadKey2 is active, read access is allowed.
- BASICCARDFILELOCKINFO.Forbidden
Read access is forbidden.

Byte 2: Write lock condition:

- BASICCARDFILELOCKINFO.Allowed
Write access is allowed
- BASICCARDFILELOCKINFO.Keyed1
Write access is forbidden with exception: when encryption with key WriteKey1 is active, write access is allowed.
- BASICCARDFILELOCKINFO.Keyed2
Write access is forbidden with exception: when encryption with key WriteKey1 or WriteKey2 is active, write access is allowed.

- `BASICCARDFILELOCKINFO.Forbidden`
Write access is forbidden.

Byte 3: Custom lock condition:

- `BASICCARDFILECUSTOMLOCKINFO.CAllowed`
No custom lock set
- `BASICCARDFILECUSTOMLOCKINFO.CLocked`
Custom lock set. File must be unlocked from inside card before it could be accessed.
- `BASICCARDFILECUSTOMLOCKINFO.CUnlocked`
Custom lock set, but file was unlocked from inside card, so file could be accessed till next card reset.

Byte 4: `ReadKey1` (see Byte 1)

Byte 5: `ReadKey2` (see Byte 1)

Byte 6: `WriteKey1` (see Byte 2)

Byte 7: `WriteKey2` (see Byte 2)

Parameter:

Number	Name	Directio	Description
1	Path	In	File or directory to set

Error values available by `LastErr`: (see `LastErr`)

Value	Description
<code>BASICCARDERROR.NOERROR</code>	No error, function call successful
<code>BASICCARDERROR.BCI_ERROR_PARM</code>	Invalid parameter passed to function call
<code>BASICCARDERROR.BCI_ERROR_TRANS_FAILED</code>	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (<code>FileErr<>0</code>)

See also: `CustomLock`, `FileLock2`, `ReadLock`, `WriteLock`

Supported cards: `ZC2.X`, `ZC3.X`

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim SW1SW2 as Long
Dim FileErr as Long
Dim Error as Long
Dim Data as Variant
Dim ReadLock as Byte
Dim WriteLock as Byte
Dim CustomLock as Byte
Dim ReadKey1 as Byte
Dim ReadKey2 as Byte
Dim WriteKey1 as Byte
Dim WriteKey2 as Byte
....
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
Data=zc.FileLock("\Test") ` Get lock condition for file \Test
Error=zc.LastErr
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if
ReadLock=Data(1)
WriteLock=Data(2)
```

3. Reference

```
CustomLock=Data(3)
ReadKey1=Data(4)
ReadKey2=Data(5)
WriteKey1=Data(6)
WriteKey2=Data(7)
```

3.1.3.23 FileLock2

Get/Set lock information of file or directory. This is basic oriented version of FileLock

Syntax: (Property, Read/Write)

Object: **FileLock2**(Path as String) as String

Content:

Get: Lock condition as text. Examples:

- "LOCK"
Custom lock is set
- "READ UNLOCK WRITE UNLOCK"
Read and write access permitted
- "READ LOCK WRITE LOCK"
Read and write access forbidden
- "READ UNLOCK WRITE LOCK"
Read access allowed, write access forbidden
- "READ LOCK KEY=1, 5 WRITE LOCK"
Read access allowed but only when encryption with key 1 or key 5 is active. Write access forbidden.

Set:

```
"Read Lock [Key= k1[, k2]]"      or
"Read Unlock"                    or
"Write Lock [Key= k1[, k2]]"     or
"Write Unlock"                   or
"Read Write Lock [Key= k1[, k2]]" or
"Read Write Unlock"              or
"Lock"                            or
"Unlock"
```

Note: Alone **Lock** and **Unlock** apply to Custom lock and unlock.

Parameter:

Number	Name	Direction	Description
1	Path	In	File or directory to set

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: FileLock, ReadLock, WriteLock, CustomLock

Supported cards: ZC2.X, ZC3.X

3.1.3.24 FilePrint

Print string at current position to file. This function just calls FileWrite.

Syntax: (Method)

Object.FilePrint(FileNum as Long, Data as String)

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file.
2	Data	In	string to write.

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: FilePut, FilePutPos, FileWrite, FileWriteLong, FileWriteSingle, FileWriteString, FileWriteUser, FilePrint

Supported cards: ZC2.X, ZC3.X

3.1.3.25 FilePut

Put content of Data to open file at current file position.

Syntax: (Method)

Object.FilePut(FileNum as Long, Data as Variant)

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file
2	Data	In	Binary Data write to file (as array of bytes)

Error values: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

3. Reference

See also: FilePutPos, FileGet, FileGetPos

Supported cards: ZC2.X, ZC3.X

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim SW1SW2 as Long
Dim FileErr as Long
Dim Error as Long
Dim FileNum as Long
Dim Data(1 To 10) as Byte
Dim i as Long
....
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
REM Open file \Test in binary mode (put, get) for reading
FileNum=zc.OpenFile2("\Test", "FOR BINARY ACCESS WRITE")
Error=zc.LastErr
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if
REM set data to write
Data(1)=1
Data(2)=2
Data(3)=3
Data(4)=4
Data(5)=5
Data(6)=6
Data(7)=7
Data(8)=8
Data(9)=9
Data(10)=10
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
REM Write 10 bytes to file
Error=zc.FilePut(FileNum,Data)
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
REM Close file
Error=zc.CloseFile(FileNum)
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if
```

3.1.3.26 FilePutPos

Put content of Data to open file at specified file position.

Syntax: (Method)

Object.FilePutPos(FileNum as Long, Pos as Long, Data as Variant)

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file

2	Pos	In	Requested write position
3	Data	In	Binary Data write to file (as array of bytes)

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: FilePut, FilePutPos, FileWrite, FileWriteLong, FileWriteSingle, FileWriteString, FileWriteUser, FilePrint

Supported cards: ZC2.X, ZC3.X

Example: see FilePut

3.1.3.27 FileRead

Read the content of file at current position.

Syntax: (Method)

Object.FileRead (FileNum as Long, Len as Integer) as Variant

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file
2	Len	In	Number of bytes to read

Return values:

Data read from file as array of byte stored in variant

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: FileGet, FileGetPos, FileRead, FileReadLine, FileReadLong, FileReadSingle, FileReadString, FileReadUser, FileReadUser2, OpenFile, OpenFile2

Supported cards: ZC2.X, ZC3.X

3.1.3.28 FileReadLine

Read line from file at current position

Syntax: (Method)

3. Reference

Object.FileReadLine(FileNum as Long) as String

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file

Return values:

Data (line) read from file.

Note: Data read must not include binary 0.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)
BASICCARDERROR.BCI_ERROR_OVERFLOW	Data exceeds buffer size

See also: FileGet, FileGetPos, FileRead, FileReadLine, FileReadLong, FileReadSingle, FileReadString, FileReadUser, FileReadUser2, OpenFile, OpenFile2

Supported cards: ZC2.X, ZC3.X

3.1.3.29 FileReadLong

Read long value from file at current position.

Syntax: (Method)

Object.FileReadLong (FileNum as Long) as Long

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file

Return values:

Long value read from file.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: FileGet, FileGetPos, FileRead, FileReadLine, FileReadLong, FileReadSingle, FileReadString, FileReadUser, FileReadUser2, OpenFile, OpenFile2.

Supported cards: ZC2.X, ZC3.X

3.1.3.30 FileReadSingle

Read Single value from file at current position.

Syntax: (Method)

Object.FileReadSingle(FileNum as Long) as Single

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file

Return values:

Float value read from file.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: FileGet, FileGetPos, FileRead, FileReadLine, FileReadLong, FileReadSingle, FileReadString, FileReadUser, FileReadUser2, OpenFile, OpenFile2

Supported cards: ZC2.X, ZC3.X

3.1.3.31 FileReadString

Read a string from a file at current position.

Syntax: (Method)

Object.FileReadString (FileNum as Long) as String

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file

Return values:

String value read from file.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)
BASICCARDERROR.BCI_ERROR_OVERFLOW	String exceeds buffer size

See also: FileGet, FileGetPos, FileRead, FileReadLine, FileReadLong, FileReadSingle, FileReadString, FileReadUser, FileReadUser2, OpenFile, OpenFile2

Supported cards: ZC2.X, ZC3.X

3. Reference

3.1.3.32 FileReadUser

Read user type data from a file at current position to Data.

Syntax: (Method)

Object.FileReadUser (FileNum as Long) as Variant

Parameter:

Number	Name	Direction	Description
4	FileNum	In	Number of file

Return values:

User binary data read from file as byte array stored in variant.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)
BASICCARDERROR.BCI_ERROR_OVERFLOW	Data exceeds buffer size

See also: FileGet, FileGetPos, FileRead, FileReadLine, FileReadLong, FileReadSingle, FileReadString, FileReadUser, FileReadUser2, OpenFile

Supported cards: ZC2.X, ZC3.X

3.1.3.33 FileReadUser2

Read block from a file at current position.

Note: In opposition to FileReadUser it's necessary to put length which match data object stored in file.

Syntax: (Method)

Object.FileReadUser2 (FileNum as Long, Length as Integer) as Variant

Parameter:

Number	Name	Direction	Description
4	FileNum	In	Number of file
6	Length	In	Number of bytes requested

Return values:

User binary data read from file as byte array stored in variant.

Value	Description
ZCBCI_NOERROR	No error, function call successful
ZCBCI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: FileGet, FileGetPos, FileRead, FileReadLine, FileReadLong, FileReadSingle, FileReadString, FileReadUser, FileReadUser2, OpenFile, OpenFile2

Supported cards: ZC2.X, ZC3.X

3.1.3.34 FileWrite

Write data at current position to file. Unlike to FileWriteUser data is not prefixed with type byte, before written to file.

Syntax: (Method)

Object.FileWrite (FileNum as Long, Data as Variant)

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file
2	Data	In	Binary data to write as array of bytes.

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: FilePut, FilePutPos, FileWrite, FileWriteLong, FileWriteSingle, FileWriteString, FileWriteUser, FilePrint

Supported cards: ZC2.X, ZC3.X

3.1.3.35 FileWriteLong

Write long value at current position to file.

Syntax: (Method)

Object.FileWriteLong (FileNum as Long, Val as Long)

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file
2	Val	In	Long value to write

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred

3. Reference

	(FileErr<>0)
--	--------------

See also: FilePut, FilePutPos, FileWrite, FileWriteLong, FileWriteSingle, FileWriteString, FileWriteUser, FilePrint

Supported cards: ZC2.X, ZC3.X

3.1.3.36 FileWriteSingle

Write single value at current position to file.

Syntax: (Method)

Object.FileWriteSingle (FileNum as Long, Val as Single)

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file
2	Val	In	Single value to write

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: FilePut, FilePutPos, FileWrite, FileWriteLong, FileWriteSingle, FileWriteString, FileWriteUser, FilePrint

Supported cards: ZC2.X, ZC3.X

3.1.3.37 FileWriteString

Write string data at current position to file.

Syntax: (Method)

Object.FileWriteString (FileNum as Long, Val as String)

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file
2	Val	In	String to write

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call

BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)
---------------------------------------	--

See also: FilePut, FilePutPos, FileWrite, FileWriteLong, FileWriteSingle, FileWriteString, FileWriteUser, FilePrint

Supported cards: ZC2.X, ZC3.X

3.1.3.38 FileWriteUser

Write user type data at current position to file.

Syntax: (Method)

Object.FileWriteUser (FileNum as Long, Data as Variant)

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file
2	Data	In	Binary data to write as array of bytes

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: FilePut, FilePutPos, FileWrite, FileWriteLong, FileWriteSingle, FileWriteString, FileWriteUser, FilePrint

Supported cards: ZC2.X, ZC3.X

3.1.3.39 FilePos

Get/Set current file pointer position.

Syntax: (Property, Read/Write)

Object.Filepos(FileNum as Long) as Long

Content:

Current file position.

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file to query

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful

3. Reference

BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: Supported cards: ZC2.X, ZC3.X

3.1.3.40 LastErr

Return error code which occurs when last function was called.

Syntax: (Property, Read only)

Object.**LastErr** as Long

Content:

Error code. One of BASICCARDERROR.___ .

See also: Constant, SW1SW2, FileErr

3.1.3.41 Mkdir

Make directory on card.

Syntax: (Method)

Object.Mkdir(Path as String)

Parameter:

Number	Name	Direction	Description
1	Path	In	Directory name to create. For example “\Test\NewDir”

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: Rmdir

Supported cards: ZC2.X, ZC3.X

3.1.3.42 OpenFile

Open file on BasicCard.

Note: There is a difference in using FileWriteUser and FilePut or FilePutPos. Write functions do not result in the same binary file data as put/get functions. You must use put/get functions when file open mode is binary (BASICCARDFILEMODE.BINARY) or random (BASICCARDFILEMODE.RANDOM), and you must use read/write functions otherwise. See BasicCard reference manual for more details. Also see another function: OpenFile2. It does exactly the same job like OpenFile but it's easier to use.

Syntax: (Method)

Object.**OpenFile**(Path as String, Mode as Integer, Recordlen as Long) as Long

Parameter:

Number	Name	Direction	Description
1	Path	In	File to open
2	Mode	In	Any valid combination of BASICCARDFILEMODE.., BASICCARDFILEACCESS.. and BASICCARDFILESHARE.. use 'OR' to concatenate values.
3	Recordlen	In	Record/block size used by file

Return values:

File number required to access open file.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: CloseFile

Supported cards: ZC2.X, ZC3.X

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim SW1SW2 as Long
Dim FileErr as Long
Dim Error as Long
Dim FileNum as Long
Dim Data as Variant
Dim DumpData as String
Dim i as Long
....
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
REM Open file \Test in binary mode (put, get) for reading
FileNum=zc.OpenFile("\Test", _
                    BASICCARDFILEMODE.BINARY OR _
                    BASICCARDFILEACCESS.ACCESS_READ OR _
                    BASICCARDFILESHARE.LOCKREADWRITE, _
                    0)

Error=zc.LastErr
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
REM Read 10 bytes from file
Data=zc.FileGet(FileNum,10)
Error=zc.LastErr
if Error<>0 then
    SW1SW2=zc.SW1SW2
```

3. Reference

```

    FileErr=zc.FileErr
    ` Do error handling here
end if
DumpData=""
For i=1 to 10
    DumpData=DumpData + Hex$(Data(i)) + " "
next i
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
REM Close file
Error=zc.CloseFile(FileNum)
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if

```

3.1.3.43 OpenFile2

Open file on BasicCard.

Note: There is a difference in using FileWriteUser and FilePut or FilePutPos. Write functions do not result in same binary file data as put functions. You must use put/get functions when file open mode is binary or random and you must use read/write functions otherwise. See BasicCard reference manual for more details.

OpenFile2 makes the same job like OpenFile but it's more easy to use in Visual Basic

Syntax: (method)

Object.**OpenFile2**(Path, mode_access_lock_len as String) as Long

mode_access_lock_len = “[**For** mode][**Access** access][lock][**Len**=recordlen]“

Parameter:

Number	Name	Direction	Description
1	Path	In	File to open
2	Mode	In	Input, Output, Append, Binary, Random
2	Access		Read, Write, or Read Write
2	Lock		Shared, Lock Read, Lock Write, Lock Read Write
2	Recordlen	In	Record/block size used by file (required for Random)

Return values:

File number required to access open file.

Error values: (see LastErr)

Value	Description
ZCBCI_NOERROR	No error, function call successful
ZCBCI_ERROR_PARM	Invalid parameter passed to function call
ZCBCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: CloseFile, OpenFile

Supported cards: ZC2.X, ZC3.X

Example:

```
Dim zc as New zcBasicCard
```

```

Dim Error as Long
Dim SW1SW2 as Long
Dim FileErr as Long
Dim Error as Long
Dim FileNum as Long
Dim Data(1 To 10) as Byte
Dim i as Long
....
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
REM Open file \Test in binary mode (put, get) for reading
FileNum=zc.OpenFile2("\Test", "FOR BINARY ACCESS WRITE")
Error=zc.LastErr
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if
REM set data to write
Data(1)=1
Data(2)=2
Data(3)=3
Data(4)=4
Data(5)=5
Data(6)=6
Data(7)=7
Data(8)=8
Data(9)=9
Data(10)=10
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
REM Write 10 bytes to file
Error=zc.FilePut(FileNum,Data)
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if
Call zc.ClearAllError ` Clear LastErr, SW1SW2, FileErr
REM Close file
Error=zc.CloseFile(FileNum)
if Error<>0 then
    SW1SW2=zc.SW1SW2
    FileErr=zc.FileErr
    ` Do error handling here
end if

```

3.1.3.44 QueryEof

Query if end of file is reached.

Syntax: (Property, Read only)

Object.**QueryEof**(FileNum as Long) as Boolean

Content:

TRUE if end of file reached, otherwise FALSE

Parameter:

Number	Name	Direction	Description
1	FileNum	In	Number of file to query

Error values available by LastErr: (see LastErr)

3. Reference

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: FileLength

Supported cards: ZC2.X, ZC3.X

3.1.3.45 Param1, Param2, ...,Param12

Property used as parameter in Transaction, Transaction2. Use this property as a parameter when you need to call user command. Following types are supported: **Byte, Integer, Long, Single, String, String*n, array of Bytes, Empty.**

Syntax: (Property, Read/Write)

Object.**Param1** as Variant

Object.**Param2** as Variant

Object.**Param12** as Variant

Content:

Value and type of user command parameter.

Note: Visual Basic uses Unicode (2 byte) characters while BasicCard uses single byte characters. When passing Strings you must not use characters which are not inside range 0 to 255. This is because strings are often used to transfer binary data. So no translation between characters passed in and characters sent to card is done. If you pass in a string using invalid characters you will get a BASICCARDERROR.BCI_ERROR_PARM error when calling Transaction. We strongly recommend you to use an array of bytes instead of strings when passing binary data to parameters.

Error values: (see LastErr)

Note: LastErr is not changed.

Examples:

To pass a Byte parameter to/from the BasicCard

```
Dim ByteVal as Byte
zc.Param1 = ByteVal
..
Error=zc.Transaction(...
..
ByteVal=zc.Param1
```

To pass an Integer parameter to/from the BasicCard

```
Dim IntVal as Integer
zc.Param1 = IntVal
..
Error=zc.Transaction(...
..
IntVal=zc.Param1
```

To pass a Long parameter to/from the BasicCard

```
Dim LongVal as Long
zc.Param1 = LongVal
..
Error=zc.Transaction(...
```

```
..
LongVal=zc.Param1
```

To pass a Single parameter to/from the BasicCard

```
Dim SVal as Single
zc.Param1 = SVal
..
Error=zc.Transaction(...
..
SVal=zc.Param1
```

To pass a String parameter to/from the BasicCard

Note: Only the last parameter of a command could be a String parameter.

```
Dim StrVal as String
zc.Param1 = StrVal
..
Error=zc.Transaction(...
..
StrVal=zc.Param1
```

or

```
Dim StrVal as Variant
Dim Data(1 To 5) as Byte ` array of byte
Data(1)=0
Data(2)=1
Data(3)=2
Data(4)=3
Data(5)=4
zc.Param1 = Data ` assign array of bytes including your data
..
Error=zc.Transaction(...
..
StrVal=zc.Param1 ` receive reply data as array of bytes
```

To pass a String*n parameter to/from the BasicCard

Note: Since only the last parameter of a command could be a String parameter, while others must be String*n, every String (or byte array) parameter assigned which is not the last parameter is interpreted as String*n with n equals to current length of this string. On the other hand every last parameter including a String (or byte array) is interpreted as String not as String*n (even if a String*n is assigned). So to have a String*n sent to BasicCard another parameter of type empty must be assigned to the following parameter, which then is the last parameter. Also you must always verify that the length of the String (or byte array) you assign exactly matches the length of String as declared inside your BasicCard card program.

```
Dim StrVal as String*5
zc.Param1 = StrVal
zc.Param2 = Empty
..
Error=zc.Transaction(..., 2)
..
StrVal=zc.Param1
```

or

```
Dim StrVal as Variant
Dim Data(1 To 5) as Byte ` array of byte
Data(1)=0
Data(2)=1
Data(3)=2
Data(4)=3
Data(5)=4
```

3. Reference

```
zc.Param1 = Data ` assign array of bytes including your data
..
Error=zc.Transaction(..., 2)
..
StrVal=zc.Param1 ` receive reply data as array of bytes
```

3.1.3.46 ReadEeprom

BasicCard READ EEPROM command (Cla C0, Ins 08). Read Length bytes from eeprom starting at address start.

Note: This command is required when loading an application (image) into a card. There is no use for this function for normal applications. You should not use it directly access the EEPROM memory of your card. This function could not be used if the card is set to state RUN.

Syntax: (Method)

Object.**ReadEeprom**(Start as Long, Length as Integer) as Variant

Parameter:

Number	Name	Direction	Description
1	Start	In	Eeprom address to start read
2	Length	In	Number of bytes to read

Return values:

Data read from eeprom as array of bytes stored in variant.

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_OVERFLOW	Too much parameter data
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.
BASICCARDERROR.BCI_ERROR_MISSING_PARM	One or more return parameter is missing

See also: WriteEeprom, ClearEeprom, EepromCRC, EepromSize

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.1.3.47 ReadKeyFile

Read ZCBasic key file as generated by “keygen” to use included keys with StartEncryption command.

Syntax: (Method)

Object.**ReadKeyFile**(KeyFile as String)

Parameter:

Number	Name	Direction	Description
1	KeyFile	In	Name of keyfile (for example “c:\BasicCrD\MyProj\keys.bas”)

Error values available by LastErr: (see LastErr)

Value	Description
-------	-------------

BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_INVALID_FILE	Invalid key file

See also: SetKey, SetPoly

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.1.3.48 ReadLock

Get/Set Read lock information of file or directory.

Syntax: (Property, Read/Write)

Object: **ReadLock**(Path as String) as Variant

Content:

ReadLock as array 1 to 3 of byte. With:

Byte 1: Read lock condition:

- BASICCARDFILELOCKINFO.Allowed
Read access is allowed
- BASICCARDFILELOCKINFO.Keyed1
Read access is forbidden with exception when encryption with key ReadKey1 is active read access is allowed.
- BASICCARDFILELOCKINFO.Keyed2
Read access is forbidden with exception when encryption with key ReadKey1 or ReadKey2 is active read access is allowed.
- BASICCARDFILELOCKINFO.Forbidden
Read access is forbidden.

Byte 2: ReadKey1

Byte 3: ReadKey2

Parameter:

Number	Name	Direction	Description
1	Path	In	File or directory to set

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: CustomLock, FileLock, FileLock2, WriteLock

Supported cards: ZC2.X, ZC3.X

3.1.3.49 Rename

Rename or move a file or directory.

Syntax: (Method)

Object: **Rename**(OldPath as String, NewPath as String)

3. Reference

Parameter:

Number	Name	Direction	Description
1	OldPath	In	Directory or file to rename or move.
2	NewPath	In	New directory or file name

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also:

Supported cards: ZC2.X, ZC3.X

3.1.3.50 RmDir

Remove directory from card.

Syntax: (Method)

Object.**RmDir**(Path as String)

Parameter:

Number	Name	Direction	Description
1	Path	In	Directory name to remove.

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: Mkdir

Supported cards: ZC2.X, ZC3.X

3.1.3.51 SetKey

Set encryption key for use with StartEncryption. Key set by this function must match key declared in card.

Syntax: (Property, Write only)

Object.**SetKey**(KeyNum as Integer) as Variant

Content:

Key either as String or array of Bytes. Note that size of key is either 8 or 16 (characters for String or bytes for array of Byte).

Parameter:

Number	Name	Direction	Description
1	Keynum	In	Number of key to set

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call

See also: SetPoly, ReadKeyFile

Supported cards: ZC1.X, ZC2.X, ZC3.X

Example: see StartEncryption

3.1.3.52 SetPoly

Set polynomial for encryption. Polynomial set by this function must match the one declared in card.

Syntax: (Method)

Object.**SetPoly**(polynomA as Long, polynomS as Long)

Parameter:

Number	Name	Direction	Description
1	polynomA	In	Polynomial A
2	polynomS	In	Polynomial S

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call

See also: SetKey, ReadKeyFile

Supported cards: ZC1.X, (ZC2.X), (ZC3.X)

3.1.3.53 StartEncryption

BasicCard START ENCRYPTION command (Cla C0, Ins 10). Start transparent encryption. Since encryption is transparently done by BasicCard-ActiveX you must have done a proper key setup (SetKey, SetPoly, ReadKeyFile) before calling this.

Syntax: (Method)

Object.**StartEncryption**(Algo as Integer, key as Integer, Random as Long)

Parameter:

Number	Name	Direction	Description
1	Algo	In	Encryption algorithm to use. Either BASICCARDENCRYPTION.SG_LFSR, BASICCARDENCRYPTION.SG_LFSR_CRC,

3. Reference

			BASICCARDENCRYPTIONDES or BASICCARDENCRYPTION.DES3
2	Key	In	Number of encryption key to use
3	Random	In	Random number used in conjunction with card random number as initial vector for encryption

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_OVERFLOW	Too much parameter data
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.

See also: EndEncryption

Supported cards: ZC1.X, ZC2.X, ZC3.X

Example:

```
Dim zc as New zcBasicCard
Dim Error as Long
Dim MyDesKey(1 To 8) as Byte
....
REM Setup des key to same value as in card
MyDesKey(1)=&HA5
MyDesKey(2)=&H17
MyDesKey(3)=&H8D
MyDesKey(4)=&H35
MyDesKey(5)=&H62
MyDesKey(6)=&H96
MyDesKey(7)=&HEF
MyDesKey(8)=&HC4
zc.SetKey(1)=MyDesKey
Error=zc.LastErr
if Error<>0 then
    \ Do error handling here
end if
....
REM Start encryption with key 1 and some simple random
Randomize
Error=zc.StartEncryption(1, MyDesKey, Int(Rnd))
if Error<>0 then
    \ Do error handling here
end if
....
Error=zc.EndEncryption
if Error<>0 then
    \ Do error handling here
end if
```

3.1.3.54 State

BasicCard GET & SET STATE command (Cla C0, Ins 00 & Ins 0C) In this case only card state (BASICCARDSTATE.LOAD, BASICCARDSTATE.TEST or BASICCARDSTATE.RUN)

Note: This command is required when loading an application (image) into a card. There is no use for this function for normal applications.

Syntax: (Property, Read/Write)

Object.State as Integer

Content:

Card state. Either BASICCARDSTATE.LOAD, BASICCARDSTATE.TEST or BASICCARDSTATE.RUN

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.

See also: Version

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.1.3.55 SW1SW2

Return command status word.

Syntax: (Property, Read only)

Object.SW1SW2 as Long

Content:

Status Word. This is the status word as returned by BasicCard. Normally with no error condition this is either BASICCARDSW1SW2.CommandOK or in the range of BASICCARDSW1SW2.LeWarning to BASICCARDSW1SW2.LeWarning + 255. You only need to check this if BASICCARDERROR.BCI_ERROR_TRANS_FAILED is returned by LastErr.

See also: Constant, LastErr, FileErr

3.1.3.56 Transaction

Do transaction or in other words call a card command.

Syntax: (Method)

Object.Transaction(Cla as Integer, Ins as Integer, ParamCount as Integer)

Parameter:

Number	Name	Direction	Description
1	Cla	In	Command class byte (first of 2 byte pair given in ZCBasic Declare Command source line)
2	Ins	In	Command instruction byte (second of 2 byte pair given in ZCBasic Declare Command source line)
5	ParamCount	In	Number of command parameters

Note: For Input/Output parameters for card command use Param1, ...,Param12. ParamCount must be number of Param1,...Param12 used. Actually up to 12 parameters are supported. Please also take a look at Param1, Param2, ...,Param12

Return value: error code (see below)

Error values returned:

3. Reference

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_OVERFLOW	Too much parameter data
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.
BASICCARDERROR.BCI_ERROR_MISSING_PARM	One or more return parameter is missing

See also:Transaction2

Supported cards: ZC1.X, ZC2.X, ZC3.X

Examples:

If BasicCard command is declared as follows:

Declare Command &H20 &H00 Test1 (MyData1 as Byte, MyData2 as Long)

Then create a function like this:

```
Dim zc as New zcBasicCard
....
Function Test1 (ByRef MyData as Byte, ByRef MyData2 as Long) as Long
    REM Assign values to parameters, so data can be transmitted and
    REM data type is known
    zc.Param1=MyData
    zc.Param2=MyData2
    Test1=zc.Transaction(&H20, &H00, 2) ` Cla &H20, Ins &H00, ParamCount=2
    REM Assign return values to function parameters
    MyData=zc.Param1
    MyData2=zc.Param2
End Function
```

If BasicCard command is declared as follows:

Declare Command &H20 &H02 Test2 (MyData1 as Integer, MyData2 as String)

Then create a function like this:

```
Dim zc as New zcBasicCard
....
Function Test2 (ByRef MyData as Integer, ByRef MyData2 as String) as Long
    REM Assign values to parameters, so data can be transmitted and
    REM data type is known
    zc.Param1=MyData
    zc.Param2=MyData2
    Test2=zc.Transaction(&H20, &H02, 2) ` Cla &H20, Ins &H02, ParamCount=2
    REM Assign return values to function parameters
    MyData=zc.Param1
    MyData2=zc.Param2
End Function
```

or if using array of bytes

```
Function Test2 (ByRef MyData as Integer, ByRef MyData2() as Byte) as Long
    REM When calling this parameter assigned to MyData2
    REM must be declared using ReDim not Dim.
    REM Assign values to parameters, so data can be transmitted and
    REM data type is known
    zc.Param1=MyData
    zc.Param2=MyData2
    Test2=zc.Transaction(&H20, &H02, 2) ` Cla &H20, Ins &H02, ParamCount=2
    REM Assign return values to function parameters
    MyData=zc.Param1
```

```

    MyData2=zc.Param2
End Function

```

If BasicCard command is declared as follows:

Declare Command &H20 &H04 Test3 (MyData1 as String*5, MyData2 as String)
Then create a function like this:

```

Dim zc as New zcBasicCard
....
Function Test3 (ByRef MyData as String, ByRef MyData2 as String) as Long
    REM Trim MyData to 5 characters (String*5)
    REM This may cause " " (space) characters to be appended
    REM or characters to be truncated
    Dim XMyData as String*5
    XMyData=MyData
    REM Assign values to parameters, so data can be transmitted and
    REM data type is known
    zc.Param1=XMyData ` Assign XMyData instead of MyData to set proper type
    zc.Param2=MyData2
    Test3=zc.Transaction(&H20, &H04, 2) ` Cla &H20, Ins &H04, ParamCount=2
    REM Assign return values to function parameters
    MyData=zc.Param1
    MyData2=zc.Param2
End Function

```

or if using array of bytes

```

Function Test3 (ByRef MyData() as Byte, ByRef MyData2() as Byte) as Long
    REM When calling this parameter assigned to MyData and MyData2
    REM must be declared using ReDim not Dim.
    REM Trim MyData to 5 bytes (Preserve keeps current data)
    ReDim Preserve MyData(1 To 5) as Byte
    REM Assign values to parameters, so data can be transmitted and
    REM data type is known
    zc.Param1=MyData
    zc.Param2=MyData2
    Test3=zc.Transaction(&H20, &H04, 2) ` Cla &H20, Ins &H04, ParamCount=2
    REM Assign return values to function parameters
    MyData=zc.Param1
    MyData2=zc.Param2
End Function

```

If BasicCard command is declared as follows:

Declare Command &H20 &H06 Test4 (MyData1 as Integer, MyData2 as String*3)
Then create a function like this:

```

Dim zc as New zcBasicCard
....
Function Test4 (ByRef MyData as Integer, ByRef MyData2 as String) as Long
    REM Trim MyData2 to 3 characters (String*3)
    REM This may cause " " (space) characters to be appended
    REM or characters to be truncated
    Dim XMyData2 as String*3
    XMyData2=MyData2
    REM Assign values to parameters, so data can be transmitted and
    REM data type is known
    zc.Param1=MyData
    REM Assign XMyData2 instead of MyData2 to set proper type (length)
    zc.Param2=XMyData2
    REM If this is not used MyData2 is interpreted as String instead of
    REM String*3
    zc.Param3=Empty
    Test4=zc.Transaction(&H20, &H06, 3) ` Cla &H20, Ins &H04, ParamCount=3!!

```

3. Reference

```

REM Assign return values to function parameters
MyData=zc.Param1
MyData2=zc.Param2
End Function

```

or if using array of bytes

```

Function Test4 (ByRef MyData as Integer, ByRef MyData2() as Byte) as Long
REM When calling this parameter assigned to MyData2
REM must be declared using ReDim not Dim.
REM Trim MyData2 to 3 bytes (Preserve keeps current data)
ReDim Preserve MyData(1 To 3) as Byte
REM Assign values to parameters, so data can be transmitted and
REM data type is known
zc.Param1=MyData
zc.Param2=MyData2
REM If this is not used MyData2 is interpreted as String instead of
REM String*3
zc.Param3=Empty
Test4=zc.Transaction(&H20, &H06, 3) ` Cla &H20, Ins &H04, ParamCount=3!!
REM Assign return values to function parameters
MyData=zc.Param1
MyData2=zc.Param2
End Function

```

3.1.3.57 Transaction2

Do enhanced transaction. This function can be used similarly to Transaction, but you can specify P1, P2, Lc and Le additionally to overwrite default values.

Syntax: (method)

Object.**Transaction2**(Cla as Integer, Ins as Integer, P1 as Integer, P2 as Integer, Lc as Long, Le as Long, ParamCount as Integer)

Parameter:

Number	Name	Direction	Description
1	Cla	In	Command class byte (first of 2 byte pair given in ZCBasic Declare Command source line)
2	Ins	In	Command instruction byte (second of 2 byte pair given in ZCBasic Declare Command source line)
3	P1	In	ISO7816 P1 (set this to 0 if not needed)
4	P2	In	ISO7816 P2 (set this to 0 if not needed)
5	Lc	In	Lc (0..255) or BASICCARDLC.DEFAULT_LC if no special value is required
6	Le	In	Le (0..255), BASICCARDLE.DEFAULT_LE (if no special value is required) or BASICCARDLE.DISABLE_LE (if Le should be omitted)
7	ParamCount	In	Number of command parameters

Note: For Input/Output parameters for card command use Param1, ...,Param12. ParamCount must be number of Param1,...Param12 used. Actually 12 parameters are supported.

Return value: error code (see below)

Error values returned:

Value	Description
-------	-------------

BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_OVERFLOW	Too much parameter data
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.
BASICCARDERROR.BCI_ERROR_MISSING_PARM	One or more return parameters are missing

See also Transaction

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.1.3.58 WriteEeprom

BasicCard WRITE EEPROM command (Cla C0, Ins 06). Write Data to eeprom starting at address start.

Note: This command is required when loading an application (image) into a card. There is no use for this function for normal applications.

Syntax: (Method)

Object: **WriteEeprom**(Start as Long, Data as Variant)

Parameter:

Number	Name	Direction	Description
1	Start	In	Start address of eeprom memory to write
2	Data	In	Data to write to eeprom as array of byte stored in variant

Return value: error code (see below)

Error values returned:

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_OVERFLOW	Too much parameter data
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.

See also: ClearEeprom, ReadEeprom

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.1.3.59 WriteLock

Get/Set Write lock information of file or directory.

Syntax: (Property, Read/Write)

Object: **WriteLock**(Path as String) as Variant

Content:

WriteLock as array 1 to 3 of byte. With:

Byte 1: Write lock condition:

3. Reference

- **BASICCARDFILELOCKINFO.Allowed**
Write access is allowed
- **BASICCARDFILELOCKINFO.Keyed1**
Write access is forbidden with exception when encryption with key WriteKey1 is active read access is allowed.
- **BASICCARDFILELOCKINFO.Keyed2**
Write access is forbidden with exception when encryption with key WriteKey1 or WriteKey2 is active read access is allowed.
- **BASICCARDFILELOCKINFO.Forbidden**
Write access is forbidden.

Byte 2: WriteKey1

Byte 3: WriteKey2

Parameter:

Number	Name	Direction	Description
1	Path	In	File or directory to set

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_PARM	Invalid parameter passed to function call
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX or file error occurred (FileErr<>0)

See also: CustomLock, FileLock, FileLock2, ReadLock

Supported cards: ZC2.X, ZC3.X

3.1.3.60 Version

BasicCard GET & SET STATE command (Cla C0, Ins 00 & Ins 0C) In this case only card version.

Syntax: (Property, Read/Write)

Object.State as Integer

Content:

BasicCard version for example 0x303 if ZC3.3

Error values available by LastErr: (see LastErr)

Value	Description
BASICCARDERROR.NOERROR	No error, function call successful
BASICCARDERROR.BCI_ERROR_TRANS_FAILED	Transaction failed. SW1SW2 differs from &H9000 or &H61XX.

See also: State

Supported cards: ZC1.X, ZC2.X, ZC3.X

3.1.4 Constants

3.1.4.1 BASICCARDSTATE

For use with State.

Value (decimal)	Value (hexadecimal)	Name	Description
0	&H0	NEW	Card state NEW.
1	&H1	LOAD	Card state LOAD. State of cards when delivered to BasicCard customers. A program can be downloaded to BasicCards in state LOAD.
2	&H2	TEST	Card state TEST. Same as RUN, but card state can be changed to LOAD to download a changed program into BasicCard
3	&H3	RUN	Card state RUN. Card is closed and could not be reprogrammed anymore. Commands like ReadEeprom or WriteEeprom are disabled in this state. Card state could not be changed anymore, so RUN is the permanent state of the card. Cards delivered to final customers should be set into this state after the program is downloaded.

3.1.4.2 BASICCARDLC

For use with Transaction2

Value (decimal)	Value (hexadecimal)	Name	Description
-1	&HFFFF	DEFAULT_LC	Use default (calculated) Lc value.

3.1.4.3 BASICCARDLE

For use with Transaction2

Value (decimal)	Value (hexadecimal)	Name	Description
-1	&HFFFF	DEFAULT_LE	Use default (calculated) Le value.
-2	&HFFFE	DISABLE_LE	Do not send Le byte.

3.1.4.4 BASICCARDERROR

Error codes returned by functions or LastErr.

Value (decimal)	Value (hexadecimal)	Name	Description
0	&H0	NOERROR	No error.
16777216	&H01000000	CRI_ERROR_PARM	Invalid parameter passed to card reader function.
16777217	&H01000001	CRI_ERROR_UNKOWN_READER	Reader name passed to function is not a known card reader device.
16777218	&H01000002	CRI_ERROR_READER_BUSY	Card reader is busy or not

3. Reference

			available.
16777219	&H01000003	CRI_ERROR_PROTOCOL	A low level protocol (T=1, T=0) error has occurred when transmitting command to card.
16777220	&H01000004	CRI_ERROR_MEM	Card reader function failed to allocate memory. This may be caused by a request expecting too much data.
16777221	&H01000005	CRI_ERROR_INITFAILED	Card reader interface failed to initialize.
16777222	&H01000006	CRI_ERROR_MISC	Unknown error when using card reader interface.
16777223	&H01000007	CRI_ERROR_NO_CARD	No card inserted in selected card reader.
16777224	&H01000008	CRI_ERROR_NO_DEFAULT	No default value available.
16777225	&H01000009	CRI_CANCEL	Dialog canceled.
16777226	&H0100000A	CRI_ERROR_OVERFLOW	Buffer overflow.
16777227	&H0100000B	CRI_ERROR_NOT_SUPPORTED	Function called is not supported by this reader or service.
33554432	&H02000000	BCI_ERROR_PARM	Invalid parameter passed to BasicCard API function.
33554433	&H02000001	BCI_ERROR_MISC	Unknown error using API.
33554434	&H02000002	BCI_ERROR_MEM	BasicCard function failed to allocate memory. This may be caused by a request expecting too much data.
33554435	&H02000003	BCI_ERROR_OVERFLOW	Buffer overflow.
33554436	&H02000004	BCI_ERROR_TRANS_FAILED	Transaction failed because card returns status word (SW1SW2) or file error code (FileErr), signaling an error condition.
33554437	&H02000005	BCI_ERROR_MISSING_PARM	Necessary parameter has not been specified or set.
33554438	&H02000006	BCI_ERROR_INVALID_FILE	Invalid file.
33554439	&H02000007	BCI_ERROR_BASE_SW	If a function returns an error code in the range from BCI_ERROR_BASE_SW to BCI_ERROR_BASE_SW + 65535 you can extract an valid SW1SW2 error code by subtract BCI_ERROR_BASE_SW

			from this value.
--	--	--	------------------

3.1.4.5 BASICCARDFILEMODE

File mode for use with OpenFile.

Value (decimal)	Value (hexadecimal)	Name	Description
96	&H60	APPEND	Open file in append mode. File could be accessed by FileWrite.. functions.
32	&H20	BINARY	Open file in binary mode. File could be accessed using FilePut, FilePutPos, FileGet and FileGetPos.
48	&H30	INPUT	Open file in input mode. File could be accessed by FileRead.. functions.
64	&H40	OUTPUT	Open file in output mode. File could be accessed by FileWrite.. functions.
80	&H50	RANDOM	Open file in random mode. File could be accessed using FilePut, FilePutPos, FileGet and FileGetPos. Note: Record length must be specified!

3.1.4.6 BASICCARDFILEACCESS

File access for use with OpenFile.

Value (decimal)	Value (hexadecimal)	Name	Description
4	&H4	ACCESS_READ	Open file for reading.
8	&H8	ACCESS_WRITE	Open file for writing.
12	&HC	ACCESS_READWRITE	Open file for reading and writing.

3.1.4.7 BASICCARDFILESHARE

File sharing for use with OpenFile.

Value (decimal)	Value (hexadecimal)	Name	Description
3	&H3	SHARED	While this file is open other (card or running program) may access this file also.
2	&H2	LOCKREAD	While this file is open other (card or running program) may access this file for writing, but not for reading.

3. Reference

1	&H1	LOCKWRITE	While this file is open other (card or running program) may access this file for reading, but not for writing.
0	&H0	LOCKREADWRITE	While this file is open other (card or running program) could not access this file.

3.1.4.8 BASICCARDSW1SW2

Status word returned by card or low level reader interface.

See BasicCard manual (chapter “Status Bytes SW1 and SW2”) as included with BasicCard development kit for a description of these values.

3.1.4.9 BASICCARDFILEERROR

File errors returned by card.

See BasicCard manual (chapter “Files and Directories”) as included with BasicCard development kit for a description of these values.

3.1.4.10 BASICCARDFILEATTRIB

File attributes as returned by Attribute

Value (decimal)	Value (hexadecimal)	Name	Description
64	&H40	CardFile	It is a file (on card).
80	&H50	CardDirectory	It is a directory (on card).

3.1.4.11 BASICCARDFILELOCKINFO

File lock information used with FileLock, ReadLock and WriteLock.

Value (decimal)	Value (hexadecimal)	Name	Description
0	&H1	Allowed	File access is allowed.
1	&H2	Keyed1	File access is allowed when encryption (StartEncryption) with first in lock information specified key is active. If this encryption is not active file access is forbidden.
2	&H3	Keyed2	File access is allowed when encryption (StartEncryption) with first or second in lock information specified key is active. If this encryption is not active file access is forbidden.
3	&H4	Forbidden	File access is forbidden.

3.1.4.12 BASICCARDFILECUSTOMLOCKINFO

File lock information used with FileLock and CustomLock.

Value (decimal)	Value (hexadecimal)	Name	Description
0	&H1	CAllowed	File access is allowed, no custom lock is active for this file.
1	&H2	CUnlocked	File access is allowed, an active custom lock was unlocked from inside card.
2	&H3	CLocked	File access is forbidden, to allow access to this file the custom lock must be unlocked from inside card.

3.1.4.13 BASICCARDENCRYPTION

Encryption algorithms to be used with StartEncryption.

Value (decimal)	Value (hexadecimal)	Name	Description
17	&H11	SG_LFSR	Shrinking Generator with Linear Feedback Shift Register (Compact BasicCard only).
18	&H12	SG_LFSR_CRC	Shrinking Generator with Linear Feedback Shift Register and CRC(Compact BasicCard only). Recommended for Compact BasicCard.
33	&H21	DES	Data Encryption Standard DES (Enhanced BasicCard only).
34	&H22	DES3	Triple DES (Enhanced BasicCard only).