

# BasicCard Crypto Control

## Visual Basic Programming Manual

Version: 1.01

Date: 31.03.00 18:15

Author: Michael Petig

email: [development@zeitcontrol.de](mailto:development@zeitcontrol.de)

Web sites:

<http://www.zeitcontrol.de>

<http://www.basiccard.com>

<http://www.basiccardusa.com>

## Contents

BasicCard Crypto Control .....	1
Visual Basic Programming Manual.....	1
1. Overview .....	3
2. Developer Guide.....	4
2.1 Setup Build Environment for C/C++ Compilers .....	4
2.2 Setup Runtime Environment.....	4
2.3 Redistribute Runtime Environment .....	4
2.4 Programming Intro .....	5
2.4.1 Overview of Supported Classes .....	5
3. Reference .....	7
3.1 Class Reference .....	7
3.1.1 zcDES .....	7
3.1.2 zcDESCert .....	10
3.1.3 zcSHA.....	13
3.1.4 zcRandom .....	15
3.1.5 zcEC161 .....	16
3.2 Constants and Enumeration .....	21
3.2.1 Error Codes.....	21
3.2.2 Enumeration ZCCRYPTMODE.....	22

## 1. Overview

This document describes the BasicCard cryptographic support functions provided by our OCX for Visual Basic. This OCX is designed to be used from Microsoft Visual Basic. For use with C/C++ development environments like Microsoft Visual C++ an additional API is available. Refer to “BasicCard Crypto API” “C/C++ Programming Manual” for using this control.

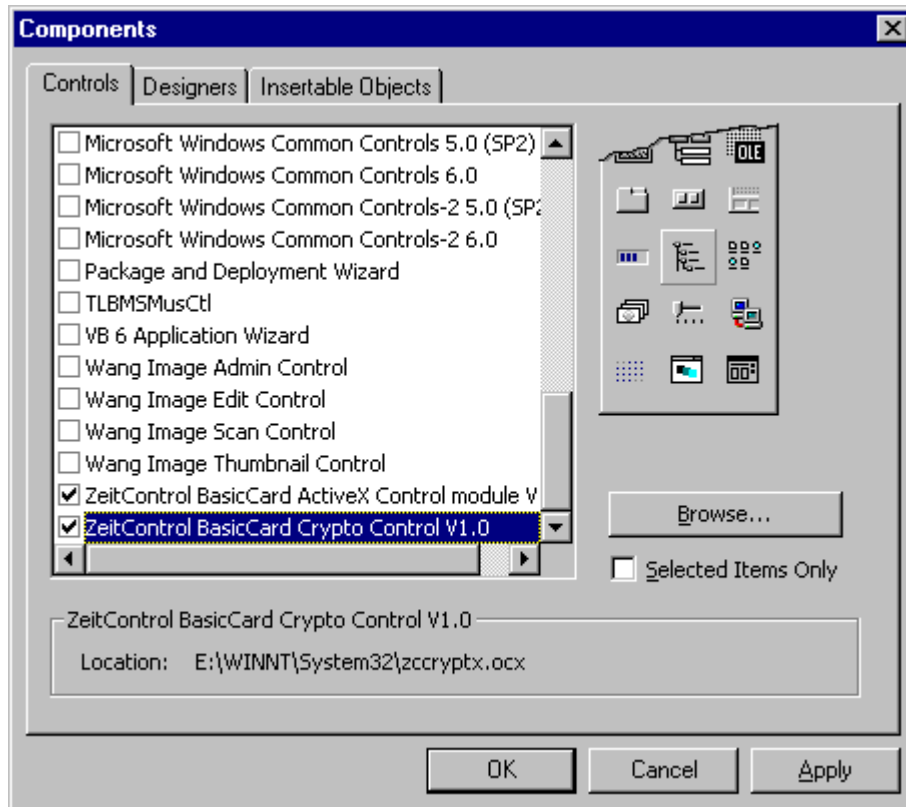
The “BasicCard Crypto Control” supports random generation, SHA, DES and EC161 (as implemented in BasicCard). It is designed to support BasicCard PC application developers by using cryptographic functions provided by BasicCard operating system and libraries. It is not designed to be used for applications unrelated to BasicCard. It may be used free without charge as long as it is used in conjunction with ZeitControl BasicCard. It is not allowed to use this API without using the BasicCard, too.

This document will describe how to use the supported cryptographic functions. We will not explain basic principles of using encryption nor will we describe BasicCard programming techniques in here. Please refer to additional literature like “Applied Cryptography” by “Bruce Schneier” for basic information about cryptography and “BasicCard, The Compact and Enhanced BasicCard” by “Tony Guilfoyle” (include in BasicCard Development Kit) for BasicCard programming and details of encryption used in BasicCard.

## 2. Developer Guide

### 2.1 Setup Build Environment for C/C++ Compilers

To install ZCCryptX.OCX you need to setup the API/OCX development package. In next step you should add ZCCryptX to your Visual Basic project. To do this choose Project/Components... from menu and select "ZeitControl BasicCard Crypto Control V1.0"



### 2.2 Setup Runtime Environment

After finishing the installation of the BasicCard API/OCX the required files are copied to your system directory. The BasicCard Crypto Control runtime currently consists of following files:

- ZCCRYPT.DLL
- ZCCRYPTX.OCX

### 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](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:

- ZCCRYPT.MSM
- ZCCRYPTX.MSM

Additionally the following Microsoft module is required:

- OLEAUT32.MSM

## 2.4 Programming Intro

### 2.4.1 Overview of Supported Classes

BasicCard Crypto Control is split into several classes. These classes are:

- **zcDES**  
Including support for DES and Triple-DES encryption.
- **zcDESCert**  
Including support for DES and Triple-DES Certificates.
- **zcSHA**  
Secure Hash functions.
- **zcRandom**  
Extended random functions for use with cryptographic functions.
- **zcEC161**  
Elliptic Curve functions as used with BasicCard since BasicCard Development Kit version 3.x.

#### 2.4.1.1 DES and Triple-DES

DES and Triple-DES is supported by zcDES class. Encryption in ECB (Electronic Code Book) and CBC (Cipher Block Chaining) mode is supported. When encrypting large amount of data chaining (CBC) should be used. In CBC mode an IV (Initial Vector) is required. This should be chosen by random and stored with encrypted message.

To use zcDES class you have to create an object of this class.

Example:

```
Dim MyDes as New zcDES
```

#### 2.4.1.2 DES and Triple-DES Certificates

Certificates using DES and Triple-DES in CBC chaining mode are supported by zcDESCert class.

To use zcDESCert class you have to create an object of this class.

Example:

```
Dim MyDesCert as New zcDESCert
```

#### 2.4.1.3 SHA

SHA is supported by zcSHA class.

To use zcSHA class you have to create an object of this class.

Example:

```
Dim MySHA as New zcSHA
```

#### 2.4.1.4 Random Generation

Normal random values as generated by standard Visual Basic function are not save to be used with encryption functions. So we have created an extended random generator. It is a difficult problem to create real secure and unpredictable random values using a standard computer. Our random generator is based on using a pool of random data which will be mixed using SHA with given seed and current random pool every time a new seed value is given to the random generator. Additionally a background thread used to collect as much "random" system data as possible to use this as seed to the random pool. Anyway we recommend to add additional random values as seed.

To use random functions of class zcRandom you have to create an object of this class. It is recommended to use a single random object, which should be declared global and initialized from Form\_Load by giving a random seed.

Example:

```
Dim MyRandom as New zcRandom
Dim Error as Long
...
```

```
Private Sub Form_Load()  
    REM Following line will cause MyRandom object to be initialized and  
    REM background thread to be started.  
    REM You should try to use something more random than "1234"  
    Error=MyRandom.Seed("1234")  
End Sub
```

### 2.4.1.5 Elliptic Curve EC161

Elliptic Curve is supported by `zcEC161` class. This class supports Elliptic Curve as implemented by BasicCard library included with BasicCard Development Kit V3.x. It is not compatible to old EC160 library as included with earlier versions of BasicCard development kit.

To use `zcEC161` class you have to create an object of this class.

Example:

```
Dim MyEC161 as New zcEC161
```

## 3. Reference

### 3.1 Class Reference

#### 3.1.1 zcDES

##### 3.1.1.1 Key

Key to be used for encryption/decryption. This could be either an array of 8 or 16 bytes. If 8 bytes are used DES encryption/decryption is performed. If 16 bytes are specified Triple-DES encryption/decryption is performed.

Syntax: (Property Read/Write)

Object.Key as Variant

It is recommended to use a single dimension array of bytes (declared using ReDim) for assigning a key to this property or requesting current active key.

Error values available by LastErr: (see LastErr).

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.NOTALLOWED	Not allowed, because already set.
ZCCRYPTERROR.ERRORPARG	Invalid parameter. Key length is neither 8 or 16 bytes.
ZCCRYPTERROR.ERRORMEM	Not enough memory-

##### 3.1.1.2 Mode

Set encryption/decryption mode. Use this property to specify chaining mode. This could either be ZCCRYPTMODE.ECB for “Electronic Code Book” or ZCCRYPTMODE.CBC for “Chipher Block Chaining”. BasicCard internal DES command uses ECB mode for encrypting/decrypting a single 64 bit (8 byte) block with each call. When encrypting large amount of data it is recommended to use chaining like CBC which could be implemented inside BasicCard by use of standard DES function with some additional Basic commands.

Note: When using CBC mode a random IV should be used and stored with encrypted message so message can be decrypted using this IV.

Syntax: (Property Read/Write)

Object.Mode as Long

Parameter: none

Error values available by LastErr: (see LastErr).

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.NOTALLOWED	Not allowed, because already set.

##### 3.1.1.3 IV

Set IV used for encryption/decryption. An IV (Initial Vector) is required if encryption/decryption is performed in CBC mode. If ECB mode is used no IV is required. If IV is not set an IV of 8 bytes filled with 0 is used. Size of IV must be always 8 bytes.

Syntax: (Property Read/Write)

Object.IV as Variant

It is recommended to use a single dimension array of 8 bytes (declared using ReDim) for assigning a IV to this property or requesting current active IV.

Error values available by LastErr: (see LastErr).

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORPARGM	Invalid parameter. IV length is not 8 bytes.
ZCCRYPTERROR.ERRORMEM	Not enough memory-

### 3.1.1.4 LastError

Use this to query if an error has occurred.

Syntax: (Property Read/Write)

Object.LastError as Long

### 3.1.1.5 Encrypt

Encrypt data. You should pass data as single dimension array of bytes. Return value should assigned to single dimension array of bytes declared using ReDim (not Dim). Length of data to encrypt must be multiple of 8 bytes.

Syntax: (Method)

Object.Encrypt(Data as Variant) as Variant.

Error values available by LastErr: (see LastErr)

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORPARGM	Invalid parameter or parameter length.
ZCCRYPTERROR.ERRORMEM	Not enough memory-

Example:

```
Dim MyDes as New zcDes
Dim Error as Long
Dim Key(1 To 8) as Byte
Dim PlainData(1 To 16) as Byte
ReDim EncData(1 To 16) as Byte
ReDim DencData1(1 To 16) as Byte
ReDim DencData2(1 To 16) as Byte
Dim i as Long
Dim DumpEnc as String
REM setup key
Key(1)=&HFE
Key(2)=&H12
Key(3)=&H45
Key(4)=&H7A
Key(5)=&H69
Key(6)=&HAA
Key(7)=&HFC
Key(8)=&H7F
REM setup data
PlainData(1)=1
PlainData(2)=2
PlainData(3)=3
PlainData(4)=4
PlainData(5)=5
PlainData(6)=6
PlainData(7)=7
PlainData(8)=8
```

```

PlainData(9)=9
PlainData(10)=10
PlainData(11)=11
PlainData(12)=12
PlainData(13)=13
PlainData(14)=14
PlainData(15)=15
PlainData(16)=16
MyDes.Key=Key
Error=MyDes.LastError
if 0<>Error then
  ` Do error handling here
end if
MyDes.Mode=ZCCRYPTMODE.CBC ` Use CBC mode
if 0<>Error then
  ` Do error handling here
end if
MyDes.IV="12345678" ` use this as IV
if 0<>Error then
  ` Do error handling here
end if
EncData=MyDes.Encrypt(PlainData)
if 0<>Error then
  ` Do error handling here
end if
REM Buid dump as string to make result visible
DumpEnc=""
for i=1 to 16
  DumpEnc=DumpEnc+Hex$(EncData(i))+ " "
next i
REM Destroy encryption context and create new decryption context
REM This is necessary iv CBC mode is used to restore original IV
REM or any time you want to use different keys.
Set MyDes=New zcDES
MyDes.Key=Key
Error=MyDes.LastError
if 0<>Error then
  ` Do error handling here
end if
MyDes.Mode=ZCCRYPTMODE.CBC ` Use CBC mode
if 0<>Error then
  ` Do error handling here
end if
MyDes.IV="12345678" ` use this as IV
if 0<>Error then
  ` Do error handling here
end if
DencData1=MyDes.Decrypt(EncData)
if 0<>Error then
  ` Do error handling here
end if
REM Now DencData1 should contain same data as PlainData
REM -----
REM This should do the same without destroying szDES object.
REM Just restore IV to required value.
MyDes.IV="12345678" ` use this as IV
if 0<>Error then
  ` Do error handling here
end if
DencData2=MyDes.Decrypt(EncData)
if 0<>Error then
  ` Do error handling here
end if

```

REM Now DencData2 should contain same data as PlainData

### 3.1.1.6 Decrypt

Decrypt data. You should pass data as single dimension array of bytes. Return value should assigned to single dimension array of bytes declared using ReDim (not Dim). Length of data to encrypt must be multiple of 8 bytes.

Syntax: (Method)

Object.Encrypt(Data as Variant) as Variant.

Error values available by LastErr: (see LastErr)

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORPARM	Invalid parameter or parameter length.
ZCCRYPTERROR.ERRORMEM	Not enough memory-

Example: see Encrypt

## 3.1.2 zcDESCert

### 3.1.2.1 Key

Key to be used for certificate. This could be either an array of 8 or 16 bytes. If 8 bytes are used DES certificate is performed. If 16 bytes are specified Triple-DES certificate is performed.

Note: Certificate is always performed using CBC mode with IV NULL (8 times 0)

Syntax: (Property Read/Write)

Object.Key as Variant

It is recommended to use a single dimension array of bytes (declared using ReDim) for assigning a key to this property or requesting current active key.

Error values available by LastErr: (see LastErr).

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.NOTALLOWED	Not allowed, because already set.
ZCCRYPTERROR.ERRORPARM	Invalid parameter. Key length is neither 8 or 16 bytes.
ZCCRYPTERROR.ERRORMEM	Not enough memory-

### 3.1.2.2 LastError

Use this to query if an error has occurred.

Syntax: (Property Read/Write)

Object.LastError as Long

### 3.1.2.3 Once

Create a certificate using a single method call. This should be used to certificate small amounts of data. When required to certificate big amounts of data Start, Update and End should be used.

Syntax: (Method)

Object.Once(Data as Variant) as Variant.

Note: It is recommended to pass data to certificate as one dimensional array of bytes.

Error values available by LastErr: (see LastErr).

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORPARG	Invalid data passed to method.
ZCCRYPTERROR.ERRORMEM	Not enough memory-

Example:

```
Dim MyDesCert as New zcDESCert
Dim Error as Long
Dim Key(1 To 8) as Byte
Dim Data(1 To 7) as Byte
ReDim Cert(1 To 8) as Byte
Dim DumpCert as String
REM setup key
Key(1)=&HFE
Key(2)=&H12
Key(3)=&H45
Key(4)=&H7A
Key(5)=&H69
Key(6)=&HAA
Key(7)=&HFC
Key(8)=&H7F
REM setup data
Data(1)=1
Data(2)=2
Data(3)=3
Data(4)=4
Data(5)=5
Data(6)=6
Data(7)=7
MyDesCert.Key=Key
Error=MyDesCert.LastError
if 0<>Error then
  ` Do error handling here
end if
Cert=MyDesCert.Once(Data)
Error=MyDesCert.LastError
if 0<>Error then
  ` Do error handling here
end if
REM Buid dump as string to make result visible
DumpCert=""
for i=1 to 8
  DumpCert=DumpCert+Hex$(Cert(i))+ " "
next i
```

### 3.1.2.4 Start

Start new certificate.

Syntax: (Method)

Object.Start() as Long.

Return value: error code (see below)

Error values returned:

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

ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORMISSINGPARM	Key is not set.

### 3.1.2.5 Update

Update certificate or in other word append data to certificate.

Syntax: (Method)

Object.Update(Data as Variant) as Long.

Note: It is recommended to pass data to Update as one dimensional array of bytes.

Return value: error code (see below)

Error values returned:

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORMISSINGPARM	Key is not set.
ZCCRYPTERROR.ERRORNOTALLOWED	Start was not called before.

### 3.1.2.6 End

End certificate started with Start and get result.

Syntax: (Method)

Object.End() as Variant.

Return value: Certificate as array of bytes (1 To 8).

Error values available by LastErr: (see LastErr)

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORMEM	Not enough memory available.
ZCCRYPTERROR.ERRORNOTALLOWED	Start was not called before.

Example:

```
Dim MyDesCert as New zcDESCert
Dim Error as Long
Dim Key(1 To 8) as Byte
Dim Data(1 To 7) as Byte
ReDim Cert(1 To 8) as Byte
Dim DumpCert as String
REM setup key
Key(1)=&HFE
Key(2)=&H12
Key(3)=&H45
Key(4)=&H7A
Key(5)=&H69
Key(6)=&HAA
Key(7)=&HFC
Key(8)=&H7F
REM setup data
Data(1)=1
Data(2)=2
Data(3)=3
Data(4)=4
Data(5)=5
```

```

Data(6)=6
Data(7)=7
MyDesCert.Key=Key
Error=MyDesCert.LastError
if 0<>Error then
    ` Do error handling here
end if
Error=MyDesCert.Start()
if 0<>Error then
    ` Do error handling here
end if
Error=MyDesCert.Update(Data)
if 0<>Error then
    ` Do error handling here
end if
REM Append data second time
REM Update can be called several times.
Error=MyDesCert.Update(Data)
if 0<>Error then
    ` Do error handling here
end if
Cert=MyDesCert.End()
Error=MyDesCert.LastError
if 0<>Error then
    ` Do error handling here
end if
REM Buid dump as string to make result visible
DumpCert=""
for i=1 to 8
    DumpCert=DumpCert+Hex$(Cert(i))+ " "
next i

```

### 3.1.3 zcSHA

#### 3.1.3.1 Hash

This property contents is the resulting hash as array (1 To 20) of byte.

Syntax: (Property Read)

Object.Hash as Variant

Parameter: none

Return value: 20 byte hash as array of bytes (1 To 20) stored in array. Assign this to array of bytes declared using ReDim (not Dim).

#### 3.1.3.2 LastError

Use this to query if an error has occurred.

Syntax: (Property Read/Write)

Object.LastError as Long

#### 3.1.3.3 Once

Create a hash using a single method call. This should be used to hash small amounts of data. When required to hash big amounts of data Start, Append and End should be used.

Syntax: (Method)

Object.Once(Data as Variant) as Variant.

Note: It is recommended to pass data to certificate as one dimensional array of bytes.

Return value: error code (see below)

Error values returned:

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORPARG	Invalid data passed to method.
ZCCRYPTERROR.ERRORMEM	Not enough memory-

Example:

```
Dim MySha as New zcSHA
Dim Error as Long
Dim Data(1 To 7) as Byte
ReDim Hash(1 To 20) as Byte
Dim DumpHash as String
REM setup data
Data(1)=1
Data(2)=2
Data(3)=3
Data(4)=4
Data(5)=5
Data(6)=6
Data(7)=7
Error=MySha.Once(Data)
if 0<>Error then
    ` Do error handling here
end if
Hash=MySha.Hash
REM Buid dump as string to make result visible
DumpHash=""
for i=1 to 20
    DumpHash=DumpHash+Hex$(Hash(i))+ " "
next i
```

### 3.1.3.4 Start

Start new hash.

Syntax: (Method)

Object.Start() as Long.

Return value: error code (see below)

Error values returned:

Value	Description
ZCCRYPTERROR.NOERROR	No error.

### 3.1.3.5 Append

Append data to hash.

Syntax: (Method)

Object.Append(Data as Variant) as Long.

Note: It is recommended to pass data to Update as one dimensional array of bytes.

Return value: error code (see below)

Error values returned:

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

ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORPARM	Invalid parameter passed to function.

### 3.1.3.6 End

End hash started with Start. Result is available in Hash property.

Syntax: (Method)

Object.End() as Long.

Return value: error code (see below)

Error values returned:

Value	Description
ZCCRYPTERROR.NOERROR	No error.

Example:

```
Dim MySha as New zcSHA
Dim Error as Long
Dim Data(1 To 7) as Byte
ReDim Hash(1 To 20) as Byte
Dim DumpHash as String
REM setup data
Data(1)=1
Data(2)=2
Data(3)=3
Data(4)=4
Data(5)=5
Data(6)=6
Data(7)=7
Error=MySha.Start()
if 0<>Error then
    ` Do error handling here
end if
Error=MySha.Append(Data)
if 0<>Error then
    ` Do error handling here
end if
Error=MySha.End()
if 0<>Error then
    ` Do error handling here
end if
Hash=MySha.Hash
REM Buid dump as string to make result visible
DumpHash=""
for i=1 to 20
    DumpHash=DumpHash+Hex$(Hash(i))+ " "
next i
```

## 3.1.4 zcRandom

### 3.1.4.1 LastError

Use this to query if an error has occurred.

Syntax: (Property Read/Write)

Object.LastError as Long

### 3.1.4.2 Seed

Seed random generator. You should call this whenever real good random is available to increase security of random out put created.

Syntax: (Method)

Object.Seed(Data as Variant)

Note: A lot of different data type could be passed in. When using strings keep in mind character must be in range from 0 to 255.

Return value: error code (see below)

Error values returned:

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORPARM	Invalid parameter passed to function.

### 3.1.4.3 GetRandom

Get random data.

Syntax: (Method)

Object.GetRandom(Length as Long) as Variant

Parameter: Requested length (in bytes) of random data (max 40).

Return value: Random data as array of bytes (1 To Length).

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORPARM	Length > 40.

Example:

```
Dim MyRandom as New zcRandom
Dim Error as Long
...
Private Sub Form_Load()
    REM Following line will cause MyRandom object to be initialized and
    REM background thread to be started.
    REM You should try to use something more random than "1234"
    Error=MyRandom.Seed("1234")
End Sub
....
ReDim RandomData(1 To 5) as Byte
RandomData=MyRandom.GetRandom(5)
Error=MyRandom.LastError
if 0<>Error then
    ` Do error handling here
end if
```

## 3.1.5 zcEC161

### 3.1.5.1 Curve

Setup curve by use of a predefined BasicCard curve.

Syntax: (Property Read/Write)

Object.Curve as Long

Note: Assigned value must be in range from 1 to 3.

Parameter: none

Return values:

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.NOTALLOWED	Curve already set.

Example:

```
Dim MyEC161 as New zcEC161
Dim Error as Long
....
MyEC161.Curve=1
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if
```

### 3.1.5.2 CurveParams

Setup curve by use of a predefined BasicCard curve. This function could be used when curve parameters (stored in EC161Params) are loaded from BasicCard. For this purpose you can use string or array of bytes as parameter type when retrieving parameters from BasicCard through BasicCard OCX.

Syntax: (Property Read/Write)

Object.CurveParams as Variant.

Note: CurveParams include data as array of bytes. When assigning to this property you may also use a string including parameters as binary data. Binary format is same as used in BasicCard:

```
Byte 1: a
Byte 2 to 22: b
Byte 23 to 43: r
Byte 44: k
Byte 45 to 65: Gx
Byte 66: Gy
```

Parameter: none

Return values:

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORPARM	Invalid parameter.
ZCCRYPTERROR.ERRORMEM	Not enough memory
ZCCRYPTERROR.ERROREC161BADPARAMS	Bad ecc parameters (*pParams)

Example: (compatible to BasicCard EC-161 example)

```
Dim zc as New zcBasicCard
Dim MyEC161 as New zcEC161
Dim Error as Long
ReDim MyCurveParams(1 To 66) as Byte
....
Private Function GetCurve(ByRef CurveParms() as Byte) as Long
...REM Call get curve command in EC-161 example card
    zc.Param1=CurveParms
```

```

...GetCurve=zc.Transaction(&H20, &H0, 1)
...CurveParms=zc.Param1
End Function
...
REM Read params from card
Error=GetCurve(MyCurveParams)
if Error<>0 then
  ` Do error handling here
end if
REM Set params
MyEC161.CurveParams=MyCurveParams
Error=zc.LastErr
if Error<>0 then
  ` Do error handling here
end if

```

### 3.1.5.3 LastError

Use this to query if an error has occurred.

Syntax: (Property Read/Write)

Object.LastError as Long

### 3.1.5.4 PublicKey

Get public key related to private key set by PrivateKey property (or generated by GenerateKeyPair).

Syntax: (Property Read)

Object.PublicKey as Variant

Note: Public key is returned as array of bytes (1 To 22).

Parameter: none

Error values available by LastErr: (see LastErr)

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORNOTALLOWED	Private key is not set..
ZCCRYPTERROR.ERRORREC161BADPARAMS	Bad ecc parameters.

### 3.1.5.5 PrivateKey

Get or set private key. When private key is set related public key can be requested by PublicKey property.

Syntax: (Property Read/Write)

Object.PrivateKey as Variant

Note: Public key is returned as array of bytes (1 To 21). To set PrivateKey a one dimensional array of bytes should be used.

Parameter: none

Error values available by LastErr: (see LastErr)

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORNOTALLOWED	Private key is not set..
ZCCRYPTERROR.ERRORREC161BADPARAMS	Bad ecc parameters.

### 3.1.5.6 GenerateKeyPair

Generate a new key pair (private key and public key).

Note: To use this function uses zcRandom class for generating key. So it is recommended to initialize a zcRandom object at form load time. Further you should pass some random data to GenerateKeyPair. This data is used to seed random generator before generating keys.

Syntax: (Method)

GenerateKeyPair(Seed as Variant) as Long

Parameter:

Number	Name	Direction	Description
1	Seed	in	Random value used to seed random generator. Note: You may pass in Empty instead, but it is strongly recommended to add some random data (which should not be generated by zcRandom) by use of this parameter.

Return values:

Value	Description
ZCCRYPTERROR.NOERROR	No error.
ZCCRYPTERROR.ERRORPARAM	Invalid parameter Seed.

### 3.1.5.7 Verify

Verify a ECC signature of 20 byte message or hash.

Syntax: (Method)

Verify(Signature as Variant, PublicKey as Variant, Message as Variant);

Parameter:

Number	Name	Direction	Description
1	Signature	in	Signature, 42 bytes as array of bytes.
2	PublicKey	in	Signer's 22 byte public key as array of bytes.
3	Message	in	20 byte message to be verified as array of bytes.

Return values:

Value	Description
ZCCRYPTERROR.NOERROR	No error, signature valid.
ZCCRYPTERROR.ERRORPARAM	Invalid parameter.
ZCCRYPTERROR.ERRORREC161BADPARAMS	Invalid public key
ZCCRYPTERROR.ERRORREC161BADSIGNATURE	Invalid signature or in other words verify failed.

Example:

```
Dim zc as New zcBasicCard
Dim MyEC161 as New zcEC161
ReDim Signature(1 To 42) as Byte
ReDim SignPubKey(1 To 22) as Byte
ReDim Message(1 To 20) as Byte
Dim Error as Long
REM Set curve matching to signers curve
```

```

MyEC161.Curve=1
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if
...
REM Signature, SignPubKey and Message must be setup here
...
Error=MyEC161.Verify(Signature, SignPubKey, Message)
if Error=0 then
    ` Signature is valid
end if

```

### 3.1.5.8 HashAndVerify

Verify a ECC signature of message by first hash message using SHA and then verify signature.

Syntax: (Method)

HashAndVerify(Signature as Variant, PublicKey as Variant, Message as Variant);

Parameter:

Number	Name	Direction	Description
1	Signature	in	Signature, 42 bytes as array of bytes.
2	PublicKey	in	Signer's 22 byte public key as array of bytes.
3	Message	in	Message of any length to be verified. Should be passed to as array of bytes.

Return values:

Value	Description
ZCCRYPTERROR.NOERROR	No error, signature valid.
ZCCRYPTERROR.ERRORPARM	Invalid parameter.
ZCCRYPTERROR.ERRORREC161BADPARAMS	Invalid public key
ZCCRYPTERROR.ERRORREC161BADSIGNATURE	Invalid signature or in other words verify failed.

Example:

```

Dim zc as New zcBasicCard
Dim MyEC161 as New zcEC161
ReDim Signature(1 To 42) as Byte
ReDim SignPubKey(1 To 22) as Byte
ReDim Message(1) as Byte
Dim Error as Long
REM Set curve matching to signers curve
MyEC161.Curve=1
Error=zc.LastErr
if Error<>0 then
    ` Do error handling here
end if
...
REM Signature, SignPubKey and Message must be setup here
...
Error=MyEC161.HashAndVerify(Signature, SignPubKey, Message)
if Error=0 then
    ` Signature is valid

```

end if

### 3.1.5.9 SessionKey

Calculate session key.

Note: You can use a session key to establish a encrypted connection between two parties. Session key generated here must therefor be used with symmetric encryption like DES.

Syntax:

SessionKey(PublicKey as Variant, Kdp as Variant) as Variant

Note: To use this function PrivateKey property must be valid.

Parameter:

Number	Name	Direction	Description
1	PublicKey	in	Other party's 22-byte public key as array of bytes.
2	Kdp	in	Key derivation parameter.

Return values:

20 byte session key as array (1 To 20) of byte.

Error values available by LastErr: (see LastErr)

Value	Description
ZCCRYPTERROR.NOERROR	No error, signature valid.
ZCCRYPTERROR.ERRORPARM	Invalid parameter (Context).
ZCCRYPTERROR.ERRORREC161BADPARAMS	Invalid key.

## 3.2 Constants and Enumeration

### 3.2.1 Error Codes

Value (Hex)	Symbolic name	Description
0	ZCCRYPTERROR.NOERROR	No error, function call succeeded.
&H03000000	ZCCRYPTERROR.ERRORPARM	Invalid parameter passed to function call.
&H03000001	ZCCRYPTERROR.ERRORMISC	Unknown, unexpected error.
&H03000002	ZCCRYPTERROR.ERRORMEM	Not enough memory.
&H03000003	ZCCRYPTERROR.ERRORNOTALLOWED	Call not allowed now, maybe some parameter not set in context.
&H03000004	ZCCRYPTERROR.ERRORMISSINGPARM	One or more parameter missing.
&H03001000	ZCCRYPTERROR.ERRORREC161BADPARAMS	Bad ECC parameters.
&H03001001	ZCCRYPTERROR.ERRORREC161BADSIGNATURE	Verify signature failed, signature is invalid.
&H03002000	ZCCRYPTERROR.ZCCRYPTERRORRNDNOTINIT	Random not initialized.
&H03002001	ZCCRYPTERROR.ZCCRYPTERRORRNDALREADYINIT	Random already initialized.

### 3.2.2 Enumeration ZCCRYPTMODE

Enumeration to select encryption chaining mode.

Value	Symbolic name	Description
1	ZCCRYPTMODE.ECB	Electronic Code Book mode
2	ZCCRYPTMODE.CBC	Chipher Block Chaining mode