

CICS® Universal Clients



# COM Automation Programming

*Version 3.1*



CICS® Universal Clients



# COM Automation Programming

*Version 3.1*

**Note!**

Before using this information and the product it supports, read the general information under “Notices” on page 101.

**Second Edition (September 1999)**

This edition applies to CICS Clients Version 3.1, program number 5648-B42, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions.

The previous edition of this book, CICS Family: OO Programming in BASIC for CICS Clients, SC33-1924-00, is still available and should be used for earlier versions of CICS Clients.

© Copyright International Business Machines Corporation 1996,1999. All rights reserved.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

---

# Contents

<b>About this book</b> . . . . .	<b>vii</b>
Who should read this book . . . . .	vii
Conventions and terminology used in this book . . . . .	vii
Prerequisite and related information . . . . .	viii
How to send your comments . . . . .	viii
Obtaining books from IBM . . . . .	ix

---

## Part 1. Client Classes—Guidance. . 1

<b>Chapter 1. Introduction to CICS OO programming</b> . . . . .	<b>3</b>
OO support in CICS Clients . . . . .	3
Programming language support. . . . .	4

<b>Chapter 2. Establishing the working environment</b> . . . . .	<b>5</b>
Environments supported . . . . .	5
The COM libraries . . . . .	5
Servers . . . . .	5
Registration . . . . .	6
Enabling the use of the COM libraries. . . . .	6

<b>Chapter 3. Using the COM classes</b> . . . . .	<b>9</b>
Object Creation and Interfaces . . . . .	9
Type Libraries and Visual Basic Intellisense . . . . .	10
Class summary . . . . .	11
Exception Handling . . . . .	12
Known Issues with Migration . . . . .	14
Making an ECI link call to CICS using Visual Basic . . . . .	14
Making an ECI link call to CICS using VBScript . . . . .	15
ECI Call Synchronization Types . . . . .	16
CICS Server Information and Connection Status . . . . .	17
ECI Link Calls within a Unit Of Work . . . . .	18
Handling COMAREAS in VB . . . . .	19
ECI Userid and Password Management. . . . .	19
Connection to CICS 3270 applications using Visual Basic . . . . .	20
Running a CICS 3270 session . . . . .	21
EPI call synchronization types . . . . .	22
CICS Server Information . . . . .	23

Using BMS Map data with EPI COM classes . . . . .	23
Connecting to CICS 3270 applications using VBScript . . . . .	25
Support for Automatic Transaction Initiation (ATI) . . . . .	25
Security Management . . . . .	26

---

## Part 2. COM classes — Reference 29

<b>Chapter 4. Buffer COM class</b> . . . . .	<b>33</b>
Interface Selection . . . . .	33
Object Creation . . . . .	33
Methods . . . . .	34
AppendString . . . . .	34
Data . . . . .	34
ExtractString . . . . .	34
InsertString . . . . .	34
Length . . . . .	34
Overlay . . . . .	35
SetData . . . . .	35
SetLength . . . . .	35
SetString . . . . .	35
String . . . . .	35

<b>Chapter 5. Connect COM class</b> . . . . .	<b>37</b>
Interface Selection . . . . .	37
Object Creation . . . . .	37
Methods . . . . .	37
AlterSecurity. . . . .	37
Cancel . . . . .	38
Changed . . . . .	38
ChangePassword . . . . .	38
Details . . . . .	39
Link . . . . .	39
MakeSecurityDefault . . . . .	40
Password . . . . .	40
ServerName . . . . .	40
ServerStatus . . . . .	40
ServerStatusText . . . . .	41
Status . . . . .	41
TranDetails . . . . .	41
UnpaddedPassword . . . . .	41
UnpaddedServerName . . . . .	42

UnpaddedUserId . . . . .	42
UserId . . . . .	42
VerifyPassword . . . . .	42

## **Chapter 6. ECI COM class . . . . . 43**

Interface Selection . . . . .	43
Object Creation . . . . .	43
Methods . . . . .	43
ErrorFormat . . . . .	43
ErrorOffset . . . . .	43
ErrorWindow . . . . .	44
ExCode . . . . .	44
ExCodeText . . . . .	44
ServerCount . . . . .	45
ServerDesc . . . . .	45
ServerName . . . . .	45
SetErrorFormat . . . . .	45

## **Chapter 7. EPI COM class . . . . . 47**

Interface Selection . . . . .	47
Object Creation . . . . .	47
Methods . . . . .	47
Diagnose . . . . .	47
ErrorFormat . . . . .	47
ErrorOffset . . . . .	48
ErrorWindow . . . . .	48
ExCode . . . . .	48
ExCodeText . . . . .	49
ServerCount . . . . .	49
ServerDesc . . . . .	49
ServerName . . . . .	49
SetErrorFormat . . . . .	49
State . . . . .	50
Terminate . . . . .	50

## **Chapter 8. Field COM class . . . . . 51**

Interface Selection . . . . .	51
Methods . . . . .	51
AppendText . . . . .	51
BackgroundColor . . . . .	51
BaseAttribute . . . . .	52
Column . . . . .	52
DataTag . . . . .	52
ForegroundColor . . . . .	52
Highlight . . . . .	53
InputProt . . . . .	53
InputType . . . . .	53
Intensity . . . . .	54
Length . . . . .	54
Position . . . . .	54

ResetDataTag . . . . .	54
Row . . . . .	54
SetBaseAttribute . . . . .	54
SetExtAttribute . . . . .	55
SetText . . . . .	55
Text . . . . .	55
TextLength . . . . .	55
Transparency . . . . .	55

## **Chapter 9. Flow COM class . . . . . 57**

Interface Selection . . . . .	57
Object Creation . . . . .	57
Methods . . . . .	57
AbendCode . . . . .	57
CallType . . . . .	57
CallTypeText . . . . .	58
Diagnose . . . . .	58
Flowid . . . . .	58
ForceReset . . . . .	58
Poll . . . . .	58
SetSyncType . . . . .	59
SetTimeout . . . . .	59
SyncType . . . . .	59
Timeout . . . . .	59
Wait . . . . .	59

## **Chapter 10. Map COM class . . . . . 61**

Interface Selection . . . . .	61
Object Creation . . . . .	61
Methods . . . . .	61
ExCode . . . . .	61
FieldByName . . . . .	62
Validate . . . . .	62

## **Chapter 11. Screen COM class . . . . . 63**

Interface Selection . . . . .	63
Methods . . . . .	63
CursorCol . . . . .	63
CursorRow . . . . .	63
Depth . . . . .	63
FieldByIndex . . . . .	64
FieldByPosition . . . . .	64
FieldCount . . . . .	64
MapName . . . . .	64
MapSetName . . . . .	64
SetAID . . . . .	65
SetCursor . . . . .	65
Width . . . . .	65

## **Chapter 12. SecAttr COM class . . . . . 67**

Interface Selection . . . . .	67
Public Methods . . . . .	67
ExpiryTime . . . . .	67
InvalidCount . . . . .	67
LastAccessTime . . . . .	68
LastVerifiedTime . . . . .	68

### **Chapter 13. SecTime COM class . . . . . 69**

Interface Selection . . . . .	69
Public Methods . . . . .	69
Day . . . . .	69
GetDate . . . . .	69
Hours . . . . .	69
Hundredths . . . . .	69
Minutes . . . . .	70
Month . . . . .	70
Seconds . . . . .	70
Year . . . . .	70

### **Chapter 14. Session COM class. . . . . 71**

Interface Selection . . . . .	71
Object Creation . . . . .	71
Methods . . . . .	71
Diagnose . . . . .	71
SetSyncType . . . . .	71
State . . . . .	72
TransId . . . . .	72

### **Chapter 15. Terminal COM class . . . . . 73**

Interface Selection . . . . .	73
Object Creation . . . . .	73
Methods . . . . .	73
AlterSecurity . . . . .	73
CCSId . . . . .	74
ChangePassword . . . . .	74
Connect . . . . .	74
Devtype . . . . .	75
Diagnose . . . . .	75
Disconnect . . . . .	75
DisconnectWithPurge . . . . .	75
DiscReason . . . . .	75
ExCode . . . . .	75
ExCodeText . . . . .	76
Install . . . . .	76
MakeSecurityDefault . . . . .	76
NetName . . . . .	76
Password . . . . .	77
Poll . . . . .	77
PollForReply . . . . .	77
QueryATI . . . . .	77

ReadTimeout . . . . .	78
ReceiveATI . . . . .	78
Screen . . . . .	78
Send . . . . .	78
ServerName . . . . .	78
SetATI . . . . .	79
SetTermDefns . . . . .	79
SignonCapability . . . . .	80
Start . . . . .	80
State . . . . .	81
TermId . . . . .	81
TransId . . . . .	81
Userid . . . . .	81
VerifyPassword . . . . .	81

### **Chapter 16. UOW COM class. . . . . 83**

Interface Selection . . . . .	83
Object Creation . . . . .	83
Methods . . . . .	83
BackOut . . . . .	83
Commit . . . . .	84
ForceReset . . . . .	84
UowId . . . . .	84

## **Part 3. Appendixes . . . . . 85**

### **Appendix A. COM Global Constants . . . . . 87**

### **Appendix B. CICS Client ECI Constants . . . . . 89**

Synchronization Types . . . . .	89
Flow status types . . . . .	89
Connection Status Codes . . . . .	89

### **Appendix C. CICS Client EPI Specific**

#### **Constants . . . . . 91**

Synchronization Types . . . . .	91
CclEPI States . . . . .	91
CclSession States . . . . .	91
CclTerminal States . . . . .	91
CclTerminal ATI States . . . . .	92
CclTerminal EndTermReasons . . . . .	92
CclTerminal Signon Types . . . . .	92
CclScreen AID key codes . . . . .	93
CclField Protected State Attributes . . . . .	94
CclField Numeric Attributes . . . . .	94
CclField Intensity Attributes . . . . .	94
CclField Modified Attributes . . . . .	94
CclField Highlight Attributes . . . . .	94
CclField Transparency Attributes . . . . .	95

CclField Color Attributes. . . . .	95	CICS Universal Clients books. . . . .	105
<b>Appendix D. Error Code References . . . .</b>	<b>97</b>	CICS Family publications . . . . .	106
<b>Notices . . . . .</b>	<b>101</b>	Book filenames. . . . .	106
Trademarks and service marks . . . . .	103	Sample configuration documents. . . . .	106
<b>Bibliography . . . . .</b>	<b>105</b>	Other publications . . . . .	107
The CICS Transaction Gateway and CICS		Viewing the online documentation . . . .	107
Universal Clients library . . . . .	105	<b>Glossary . . . . .</b>	<b>109</b>
CICS Transaction Gateway books . . . .	105	<b>Index. . . . .</b>	<b>111</b>



---

## About this book

This book describes object-oriented programming for the CICS external call interface (ECI) and CICS external presentation interface (EPI). It provides guidance on writing programs, with examples, using the classes and methods provided.

---

### Who should read this book

This book is for CICS application programmers who want to know how to use the OO classes provided in IBM CICS Universal Clients version 3.1 to develop object oriented CICS client programs.

CICS services are available to clients through the External Call Interface (ECI), the External Presentation Interface (EPI) and the External Security Interface (ESI). The CICS client COM classes allow a programmer to access the ECI and EPI interfaces in an object oriented manner.

This version supports Microsoft Visual Basic and Visual Basic for Applications and VBScript.

Details of the Component Object Model (COM) classes in this book are viewed from a Visual Basic and VBScript point of view. Although these classes should work from other development environments and languages that support COM automation these have not been tested by IBM.

The previous edition of this book, CICS Family: OO Programming in BASIC for CICS Clients, SC33-1924-00, is still available and should be used for earlier versions of CICS Clients.

---

### Conventions and terminology used in this book

Here are some of the conventions used in this book:

Filenames are shown in a monospaced font — `cuc.ini`

COM classes are shown in bold — **Buffer**

OO methods are shown in bold — **FieldCount**

Parameters are shown in italic — *serverName*

---

## Prerequisite and related information

This document assumes that you are familiar with OO concepts and the Visual Basic language, and have a reasonable understanding of the existing services that CICS provides.

For more information about using the CICS services, see *CICS Family: Client/Server Programming*.

---

## How to send your comments

Your feedback is important in helping to provide the most accurate and high-quality information. If you have any comments about this book, or any other CICS documentation:

- Visit our Web site at:

<http://www.ibm.com/ts/cics/>

and follow the **library** link to our feedback form.

Here you will find the feedback page where you can enter and submit your comments.

- Send your comments by e-mail to [idrpf@hursley.ibm.com](mailto:idrpf@hursley.ibm.com)
- Fax your comments to:

+44-1962-870229 (if you are outside the UK)  
01962-870229 (if you are in the UK)

- Mail your comments to:

Information Development  
Mail Point 095  
IBM United Kingdom Laboratories  
Hursley Park  
Winchester  
Hampshire  
SO21 2JN  
United Kingdom

Whichever method you use, ensure that you include:

- The name of the book
- The form number of the book
- If applicable, the version of the product
- The specific location of the text you are commenting on, for example, a page number or table number.

---

## Obtaining books from IBM

You can order publications through your IBM representative or the IBM branch office serving your locality.

You can check the availability of books from our Web site at:

<http://www.ibm.com/ts/cics/>

Follow the **library** link and download books as required.

Also, you can order books from the Web site at:

<http://www.elink.ibm.link.ibm/pbl/pbl>



---

# Part 1. Client Classes—Guidance

## Chapter 1. Introduction to CICS OO

<b>programming</b>	3
OO support in CICS Clients	3
Programming language support.	4

## Chapter 2. Establishing the working

<b>environment</b>	5
Environments supported	5
The COM libraries	5
Servers	5
Registration	6
Registering in-process servers	6
Registering out-of-process servers	6
Enabling the use of the COM libraries.	6

## Chapter 3. Using the COM classes

Object Creation and Interfaces	9
Type Libraries and Visual Basic Intellisense	10
Class summary	11
Exception Handling	12
Known Issues with Migration	14
Making an ECI link call to CICS using Visual Basic	14
Making an ECI link call to CICS using VBScript	15
ECI Call Synchronization Types	16
CICS Server Information and Connection Status	17
ECI Link Calls within a Unit Of Work	18
Handling COMAREAS in VB	19
ECI Userid and Password Management	19
Connection to CICS 3270 applications using Visual Basic	20
Running a CICS 3270 session	21
EPI call synchronization types	22
CICS Server Information	23
Using BMS Map data with EPI COM classes	23
Connecting to CICS 3270 applications using VBScript	25
Support for Automatic Transaction Initiation (ATI)	25
Security Management	26



---

## Chapter 1. Introduction to CICS OO programming

The CICS<sup>®</sup> family provides robust transaction processing capabilities across the major hardware platforms that IBM<sup>®</sup> offers, and also across key non-IBM platforms such as UNIX. It offers a wide range of features for supporting client/server applications, and allows the use of modern graphical interfaces for presenting information to the end user. The CICS family now supports the emerging technology for object oriented programming and offers CICS users a way of capitalizing on many of the benefits of object technology while making use of their investment in CICS skills, data and applications.

Object oriented programming allows more realistic models to be built in flexible programming languages. You can define new types or classes of objects, as well as employing a variety of structures to represent these objects.

Object oriented programming also allows you to associate more meaning with data by creating methods (member functions) that define the behavior associated with objects of a certain type, thereby capturing more of the semantics associated with the underlying data.

The hiding or encapsulating of much of the complexity of a piece of software inside a simpler external shell provides the key to reuse of code. An object defined in such a way can be used from a wide range of different applications. The provision of discrete, well defined objects can be the foundation of a library of reusable parts from which future applications can be built more quickly and cheaply. The reuse of existing parts leads to better levels of software quality as they have already been tested and used in other applications.

---

### OO support in CICS Clients

The principal communication mechanism provided on the CICS Client is the External Call Interface (ECI). The provision of an ECI class library, modelling the full function of the ECI in an object oriented way, provides the base upon which extended support has been built.

Supplied classes offer the capability of making links to servers, making calls to the CICS programs on the server, finding out status, and making use of units of work (UOW's).

The second interface available to application programmers on the CICS Client is the External Presentation Interface (EPI). Communication with 3270

## Introduction to OO programming

terminal based CICS applications is provided by classes encapsulating terminals, screens, fields and BMS maps. The EPI classes make it unnecessary to work directly with the 3270 datastream and make it easier to examine and update the contents of an output screen.

---

### Programming language support

OO libraries are provided with CICS Clients Version 3.1 for C++ and COM programmers. This book covers the COM programming topics. If you wish to program using C++, then please refer to *OO Programming in C++ for CICS Clients* where you will find a user guide based on some samples and a description of the classes.



---

## Chapter 2. Establishing the working environment

You are provided with Component Object Model (COM) Object Oriented (OO) support for CICS clients in Windows environments. This includes the COM runtimes, type libraries, the BMS map utility, and sample code.

---

### Environments supported

#### Windows NT

Windows NT Workstation version 4.00 SP3 or higher.  
Microsoft Visual Basic 5.0 or higher.  
Microsoft VBScript version 5.0 or higher.

#### Windows95 / Windows98

Microsoft Windows 95 or Microsoft Windows 98 (depending on the version supported by your CICS Client).  
Microsoft Visual Basic 5.0 or higher.  
Microsoft VBScript version 5.0 or higher.

Refer to *CICS Client Administration*, SC33-1436-00, for details of CICS server platforms supported by the CICS clients.

---

### The COM libraries

The COM libraries are automation compatible. Automation compatible means that they can be used easily from Programming languages such as Visual Basic, VBScript and Delphi. However, these libraries have only been tested with Visual Basic and VBScript and full support is not provided for other development environments.

#### Servers

The libraries are provided as in-process servers (cclicci.dll and cclicpi.dll) and out-of-process or local servers (ccloeci.exe and ccloepi.exe). Although local servers are provided they are not supported and have not been tested in a DCOM environment.

It is recommended that you use in-process servers as they provide better performance than the out-of-process servers.

## Establishing the working environment

### Registration

The COM libraries must be registered at installation time. Registration registers the COM classes, associated ProgIDs (see later) and the type libraries for each of the libraries.

Visual Basic will only use the type libraries if you register them to each Visual Basic Project. It is recommended that you do this to make full use of the the features and performance enhancements of these type libraries. See “Enabling the use of the COM libraries” for details about project enablement.

VBScript does not use type libraries.

All the COM libraries support automatic registration and de-registration and you may register in-process or out-of-process libraries.

If both the in-process and out-of-process libraries are registered, by default Visual Basic and VBScript will use the in-process versions.

#### **Registering in-process servers**

To register in-process servers use the Microsoft supplied program REGSVR32.

For example to register the ECI COM libraries issue the command  
REGSVR32 CCLIECI.DLL

to de-register issue the command  
REGSVR32 /U CCLIECI.DLL

#### **Registering out-of-process servers**

To register the out-of-process servers, run the executable and, when a window appears indicating that registration has been done, close it. For example to register the EPI libraries run the program CCLOEPI.

To unregister the out-of-process servers, run the executable with the /unregister parameter. For example to unregister the EPI libraries run the program CCLOEPI /unregister.

### Enabling the use of the COM libraries

To set up Visual Basic to use the type libraries, go to the Visual Basic Project/References... dialog and select either

IBM CICS Client EPI

or

IBM CICS Client ECI

depending on your application needs. Press OK.

If the type libraries are not listed then the COM libraries probably have not been registered. Please refer to the previous section for information on registering the COM libraries.



---

## Chapter 3. Using the COM classes

The CICS client provides COM classes for the CICS ECI and EPI on Microsoft Windows NT, Microsoft Windows 95 or Microsoft Windows 98. The interfaces could be used from any application development environment that supports COM automation. However, this book refers only to Visual Basic and VBScript. Only these environments are supported.

These COM classes can be accessed from Microsoft Visual Basic, from Visual Basic for Applications (which is provided built-in to applications such as Microsoft Excel version 5.0.), and from VBScript.

For examples of programs that use the COM classes, refer to the separate samples documentation.

---

### Object Creation and Interfaces

To talk to COM objects you have to use interfaces. The ECI and EPI COM libraries provide 2 interfaces per COM class.

The first interface is called IDispatch and is provided to support old Visual Basic applications and VBScript. A second interface, a Custom interface, is also provided for use by Visual Basic. This interface is faster than the IDispatch interface and it is recommended that you use this interface with Visual Basic. Each COM class provides an IDispatch interface and a Custom interface.

Visual Basic provides more than one way to create a COM object and select the interface to talk to that object. To create an object there are the CreateObject function and the New function. It is recommended that you use the New function to create objects in Visual Basic.

VBScript is simpler. It provides only one way to create an object, the CreateObject function, and you must use the IDispatch interface.

The following are some examples of creating COM objects

```
set eci = CreateObject("Ccl.ECI")
set eci = new CcloECI
set connection = CreateObject("Ccl.Connect")
set connection = New CcloConn
```

## COM classes

Note the two ways you can request the object class. When using `CreateObject` you specify a string called the Programmatic ID or ProgID for short. When using the `New` function you specify the Class name that is registered in the type library.

When using Visual Basic you have the choice of which interface you want to use. If you DIM your variable as `Object`, then you select the `IDispatch` interface. If you DIM your variable as the Class name then you will select the custom interface. To create a terminal object in Visual Basic you would use the code:

```
Dim Terminal as Cc10Terminal
Set Terminal = New Cc10Terminal
```

*Figure 1. Creating a terminal object in Visual Basic*

or you can combine the above into a single statement if you wish

```
Dim Terminal as New Cc10Terminal
```

When using VBScript, VBScript will automatically select the `IDispatch` interface for you. For example to create a terminal Object in VBScript you would use the code

```
Dim Terminal
Set Terminal = CreateObject("Cc1.Terminal")
```

*Figure 2. Creating a terminal Object in VBScript*

It is recommended that you:

- choose one interface type or the other.
- do not mix the object interface types in your program. This type of environment is not supported.
- select the custom interface because it should provide performance improvements.

No matter which interface you select or how the object is created, you use the objects identically in your program.

---

## Type Libraries and Visual Basic Intellisense

Type libraries add many useful features to the COM libraries. One of these is Visual Basic Intellisense. The type libraries provide Visual Basic with information so that it can help you with code completion. It prompts you with the format of the method and, where applicable, constants which might be relevant to method parameters or return values you can test for. For

example if you create a terminal object for Visual Basic as shown in Figure 1 on page 10, when you want to select a method on the terminal object, press the '.' key and you are presented with a list of available methods. Select the method and press space or open bracket and you are shown the required parameters. You can also browse the type libraries for reference information on the ECI and EPI classes by using the Visual Basic Object Browser. Select either CcIECLib for ECI classes reference or CcIEPILib for EPI classes reference information. The type libraries are embedded within the in-process library files cclieci.dll and ccliepi.dll.

---

## Class summary

The COM classes that the CICS COM servers provide are listed in the following table:

*Table 1. ECI COM classes*

COM class	Description
Buffer	Buffer used for passing data to and from a CICS server
Connection	Controls a connection to a CICS server
ECI	Provides access to a list of CICS servers configured in the client
Flow	Controls a single interaction with CICS server program
SecAttr	Provides information about security attributes (passwords)
SecTime	Provides date and time information
UOW	Coordinates a recoverable set of calls to a CICS server

*Table 2. EPI COM classes*

COM class	Description
EPI	Initializes and terminates the CICS EPI and provides access to a list of CICS servers configured in the client
Field	Provides access to a single 3270 field on a screen.
Map	Provides access to 3270 fields defined by a CICS server BMS map
Screen	Provides access to a 3270 terminal screen
Session	Controls a sequence of 3270 terminal interactions with a CICS server
Terminal	Controls a 3270 terminal connection

Samples illustrating the use of the ECI and EPI COM classes from Visual Basic are provided with the CICS client in directory xxxxxxxx\SAMPLES\VB. The following sections of this programming guide show extracts from the samples.

---

### Exception Handling

With the ECI and EPI classes there appear to be two ways to check for problems when invoking methods.

One way could be to use the `ErrorWindow` method and set it to false, then check the `ExCode` and `ExCodeText` methods after a call to see what the return codes are. This is **not** the recommended way to do it and only exists now to support backward compatibility for old applications.

The recommended way is to use the `Err` objects which Visual Basic and VBScript provide. An `Err` object contains the information about an error. Visual Basic supports `On Error Goto` and `On Error Resume` features to detect that an error has occurred. VBScript only supports the `On Error Resume Next` feature. If you use `On Error Resume Next` either in Visual Basic or VBScript, you must always enter this line before any COM object call that you expect could return an error. Visual Basic/VBScript might not reset the `Err` variable unless you do this.

The type of interface you have selected (you DIM'ed a variable as either `Object` or `classname`) will affect the value contained in the `Err.number` property. It is possible to write a generic routine that handles all values in `Err.Number` and converts them to the documented `ExCode` error codes available. The example code following shows how to achieve this.

To get full advantage of this technique, ensure that you get full information in the `Err` object. Issue the following call after creating the ECI or EPI object:

```
ECI.SetErrorFormat 1
```

or, for EPI.

```
EPI.SetErrorFormat 1
```

Figure 3 on page 13 shows how to handle errors in Visual Basic.



```

Private Sub Command1_Click()
'
' The following code assumes you have created the
' required objects first, ECI, Connect, Flow, UOW,
' Buffer
'
On Error GoTo ErrorHandler
conn.Link flow, "EC01", buf, uow
Exit Sub
ErrorHandler:
'
' Ok, the Connect call failed
' Parse the Error Number, this will work regardless of
' how the ECI objects were Dimmed
'
Dim RealError As CcIECIExceptionCodes
RealError = Err.Number And 65535 - eci.ErrorOffset
If RealError = ccITransaction Then
'
' Transaction abend, so query the Abend code
'
    AbendCode = flow.AbandCode
    If AbendCode = "AEY7" Then
        MsgBox "Invalid Userid/Password to execute CICS Program", , "CICS ECI Error"
    Else
        MsgBox "Unable to execute EC01, transaction abend:" + AbendCode, , "CICS ECI Error"
    End If
Else
    MsgBox Err.Description, , "CICS ECI Error"
End If
End Sub

```

*Figure 3. Visual Basic exception handling sample*

Figure 4 on page 14 shows error handling code for VBScript.

## COM classes

```
On Error Resume Next
con.Link flow, "EC01", buf, uow
if Err.Number <> 0 then
'
' Ok, the Connect call failed
' Parse the Error Number, this will work regardless of
' how the ECI objects were Dimmed
'
    RealError = Err.Number And 65535 - eci.ErrorOffset
'
' 13 = CclTransaction, a transaction abend.
'
    If RealError = 13 Then
'
' Transaction abend, so query the Abend code
'
        AbendCode = flow.AbandCode
        If AbendCode = "AEY7" Then
            Wscript.Echo "Invalid Userid/Password to execute CICS Program"
        Else
            Wscript.Echo "Unable to execute EC01, transaction abend:", AbendCode
        End If
    Else
        Wscript.Echo Err.Description
    End If
End If
```

*Figure 4. VBScript exception handling sample*

### Known Issues with Migration

As discussed in Exception Handling, the values in the Err.Number will vary depending on which interface and Error Format you selected. If an old application used DIM to define the variables as their class name, and checked Err.Number for the value of 8600 (other values were not available in previous releases), this check will fail.

---

## Making an ECI link call to CICS using Visual Basic

The first step is to declare object variables for the ECI interfaces to be used, usually in the General Declarations section of a Visual Basic program:

```
Dim ECI As Cc10ECI
Dim Connect As Cc10Conn
Dim Flow As Cc10Flow
Dim Buffer As Cc10Buf
Dim UOW As Cc10UOW
```

The required ECI objects are then instantiated using the Visual Basic **New** function. This can be done in the **Form\_Load** subroutine or at some later stage

in response to some user action. Note that a CclOEI object must be created first.

```
Sub ECILink_Click()
    Set ECI = New CclOEI
    Set Connect = New CclOConn
    Set Flow = New CclOFlow
    Set Buffer = New CclOBuf
```

Details of the CICS server to be used – server name (as configured in the Client initialization file) , userid and password – are supplied via the **Details** method on the Connect object. The Buffer object is initialized with some data to be sent to CICS:

```
Connect.Details "CICSNAME", "sysad", "sysad"
Buffer.SetString "Hello"
```

Now we are ready to make the call to CICS. The **Link** method takes as parameters the Flow object, the name of the CICS server program to be invoked, the Buffer object and a UOW object. In this example a null variable is supplied for the UOW parameter, so this call will not be part of a recoverable Unit Of Work. The contents of the Buffer returned from CICS are output to a Visual Basic text box "Text1":

```
Connect.Link Flow, "ECIWT0", Buffer, UOW
Text1.Text = Buffer.String
```

Finally the CICS COM objects are deleted:

```
Set Connect = Nothing
Set Flow = Nothing
Set Buffer = Nothing
End Sub
```

This example sends and receives a simple text string. In practice, the Buffer object would contain more complex data (for example a COBOL or C data structure). For binary data the **Buffer.SetData** and **Buffer.Data** methods are provided to allow the contents to be accessed as a Byte array.

A typical client application could access CICS through one or more **Connect.Link** calls and construct a 'business object' for use in end-user Basic programs. One approach to this would be to implement the 'business object' as a separate COM automation server containing the logic to process the contents of the CclOBuffer objects.

## Making an ECI link call to CICS using VBScript

This is similar to the previous section visual basic but the creating of the objects is different.

It is not necessary to DIM any variables with VBScript but it would be good programming practice to do so.

```
Dim ECI, Connect, Flow, Buffer, UOW
```

To create the objects you use the code

```
Set ECI = CreateObject("Ccl.ECI")
Set Connect = CreateObject("Ccl.Connect")
Set Flow = CreateObject("Ccl.Flow")
Set Buffer = CreateObject("Ccl.Buffer")
Set UOW = Nothing
```

If you are not going to use a UOW, you must explicitly set it to 'Nothing' in VBScript.

### ECI Call Synchronization Types

The CICS client ECI COM classes support synchronous (“blocking”) and deferred synchronous (“polling”) protocols. These classes do not support the asynchronous calls that are available in the C++ classes.

In the previous example a Flow object was used with the default synchronization type of cclSync. When this Flow object was used as the first parameter on **Connect.Link**, a synchronous link call was made to CICS. The Visual Basic program was then blocked until the reply was received from CICS. When the link call returned the reply from CICS was immediately available in the Buffer object.

To make a deferred synchronous call you use the **SetSyncType** method on the Flow object to set the Flow to cclDSync. When this Flow object is used on a **Connect.Link** call, the ECI call is made to CICS, but control returns immediately to the Visual Basic Program, and the reply from CICS must be retrieved later using the **Poll** method on the Flow object:

```
Sub ECIDsync_Click()
    Set Connect = New CclOConn
    Set Flow = New CclOFlow
    Set Buffer = New CclOBuf
    Connect.Details "CICSNAME", "sysad", "sysad"
    Flow.SetSyncType cclDSync
    Buffer.SetString "Hello"
    Connect.Link Flow, "ECIWT0", Buffer, UOW
End Sub
```

The call to CICS is now in progress. At a later stage (in response to a user action, or perhaps when the Visual Basic program has completed some other task) the **Poll** method is used on the Flow object to collect the reply from

CICS. Note that the **Poll** method requires a Buffer object as parameter if reply data is expected from CICS

```
Sub ECISReply_Click()
    If Flow.Poll(Buffer) Then
        Text1.Text = Buffer.String
    Else
        Text1.Text = "No reply from CICS yet"
    End If
End Sub
```

## CICS Server Information and Connection Status

The **ECI** COM class provides the names and descriptions of CICS servers configured in the Client initialization file. The **Connect** COM class provides methods for querying the availability of a particular CICS server.

Object variables are declared as before, this time we use **ECI**, **Connect** and **Flow** COM classes:

```
'Declare object variables
Dim ECI As Cc10ECI
Dim Connect As Cc10Conn
Dim Flow As Cc10Flow
```

On user request, the objects are created, and a list of CICS server names and their descriptions is constructed:

```
Sub ECIServers_Click()
    Dim I as Integer

    'Instantiate CICS ECI objects
    Set ECI = New Cc10ECI
    Set Connect = New Cc10Conn
    Set Flow = New Cc10Flow

    'List CICS server information
    For I = 1 To ECI.ServerCount
        List1.AddItem ECI.ServerName(I)
        List1.AddItem ECI.ServerDesc(I)
    Next
End Sub
```

A synchronous status call to the first server is made, and the results of the call displayed in a text field:

```
Connect.Details ECI.ServerName(1)
Connect.Status Flow
Text1.Text = Connect.ServerStatusText
```

### ECI Link Calls within a Unit Of Work

Using the **UOW** COM class, a number of link calls can be made to a CICS server within a single Unit of Work. Updates to recoverable resources in the CICS server can then be committed or backed out by the client program as necessary.

In this example a UOW object is created, and is used as a parameter to the **Connect.Link** calls:

```
Sub ECIStartUOW_Click()  
    'Instantiate CICS ECI objects  
    Set Connect = New Cc10Conn  
    Set Flow = New Cc10Flow  
    Set UOW = New Cc10UOW  
    Set Buffer = New Cc10Buf  
    Connect.Details "CICSNAME", "sysad", "sysad"  
End Sub  
  
Sub ECILink_Click()  
    'Set up the commarea buffer  
    Buffer.SetString Text1.Text  
    Buffer.SetLength 80  
    'Make the link call as part of a Unit of Work  
    Connect.link Flow, "ECITSQ", Buffer, UOW  
End Sub
```

After a number of link calls have been made, the **Commit** or **Backout** methods on the **Ccl UOW** interface can be used:

```
Sub Commit_Click()  
    'Commit the CICS updates  
    UOW.Commit Flow  
End Sub  
Sub Backout_Click()  
    'Backout the CICS updates  
    UOW.Backout Flow  
End Sub
```

If no UOW object is used (a NULL value is supplied on the **Connect.Link** call), each link call becomes a complete unit of work (equivalent to LINK SYNCONRETURN in the CICS server).

When you use Logical units of work, you must ensure that you backout or commit active units of work, this is particularly important at program termination. You can check if a logical unit of work is still active by checking the `uowId` method for a non-zero value.

In Visual Basic, if you Dim a UOW variable but never create the object, it is assumed to be of value **Nothing** and the Link call will therefore not associate a unit of work with the call. In VBScript, however, it is necessary to ensure explicitly that the variable is set to nothing. To do this code

```
Set UOW=Nothing
```

before making your link call.

## Handling COMAREAS in VB

A CommArea is a block of storage that contains all the information you send to and receive from the server. Because of this, you must create a CommArea that is big enough for this information. For example, you might need to send a 12 byte serial number to the server, but receive a maximum of 20 Kb back from the server; this means you must create a Commarea of size 20 Kb. To do this you could code

```
Set Buf = new CcIOBuf ' create extensible buffer object
Buf.SetString(serialNo)
Buf.setLength(20480) ' stores Nulls in the unused area
```

In the above example, Commarea is given the serial number and the buffer is increased to the required amount, but the extra area is filled with nulls. This is important as it ensures that the information transmitted to the server is kept to a minimum. The client strips off the excess nulls and only transmits the 12 bytes to the server.

---

## ECI Userid and Password Management

You are now able to do security management on Servers that support **Password Expiry Management**. Please refer to the *CICS Family: Client/Server Programming* for more information on supported servers and protocols.

To use these features you must first create a Connection object and invoke the **Details** method to associate a userid and password with the object. The two methods available are **VerifyPassword** that checks the userid and password within the connection object with the Server Security System, and **ChangePassword** that allows you to change the password at the server. If successful the connection object password is updated accordingly.

If either call is successful, you are returned a CcIOSecAttr object. This object provides access to information such as last verified time, expiry time and last access time. If, for example, you query the last verified time, you are returned a CcIOSecTime object and you may use the SecTime COM class methods to obtain the information in various formats. The following code shows the use of these various objects.

## Guide to COM CICS EPI

```
' Connection object already created called conn

on error goto pemhandler

dim SecAttr as Cc10SecAttr
dim LastVerified as Cc10SecTime
dim lvddate as Date

set SecAttr = conn.VerifySecurity
set LastVerified = SecAttr.LastVerifiedTime

lvddate = LastVerified.GetDate
strout = Format(lvddate, "hh:mm:ss, dddd, mmm d yyyy")
Text1.Text = strout

exit sub

pemhandler:

' handle a expired password here maybe
end sub
```

---

## Connection to CICS 3270 applications using Visual Basic

**Note:** The COM objects do not support DBCS fields in 3270 datastreams.

The first step is to declare object variables for the EPI interfaces to be used, usually in the General Declarations section of a Visual Basic program:

```
Dim EPI As Cc10EPI
Dim Terminal As Cc10Terminal
Dim Session As Cc10Session
Dim Screen As Cc10Screen
Dim Field As Cc10Field
```

The required EPI objects are then instantiated using the Visual Basic **New** function. This can be done in the **Form\_Load** subroutine or at a later stage in response to a user action.

The Cc10EPI object must be created first to initialize the CICS client EPI. A Cc10Terminal object can then be created, and a connection established to a specific CICS server using the **Terminal.Connect** method. The first parameter to this method is the CICS server name (as configured in the Client initialization file), the other parameters specify additional connection details (see the reference section “Connect” on page 74).



```

Sub EPICheck_Click()
    'Create CcI.EPI first to initialize EPI
    Set EPI = New CcIOEPI
    'Create a terminal object and connect to CICS
    Set Terminal = New CcIOTerminal
    Terminal.Connect "CICSNAME","", ""
    'Create a session object (defaults to synchronous)
    Set Session = New CcIOSession
End Sub

```

## Running a CICS 3270 session

Having connected a CcIOTerminal object to the required CICS server the **Terminal**, **Session**, **Screen** and **Field** COM classes are used to start a transaction on CICS and navigate through 3270 panels, accessing 3270 fields as required by the application.

The required CICS transaction is started using its 4 character transaction code. Initial transaction data can also be supplied on the **Terminal.Start** method, in this example no data is required. To access the 3270 data returned by CICS, a screen object is obtained from the terminal object, and a variety of methods can be used to obtain fields from the screen and read and update text and attributes in the fields:

```

Sub EPIScreen_Click()
    'Start CESN transaction
    Terminal.Start Session, "CESN", ""
    'Get the screen object
    Set Screen = Terminal.Screen
    'Output the text from some 3270 fields
    Set Field = Screen.FieldByIndex(5)
    List1.AddItem Field.Text
    Set Field = Screen.FieldByIndex(6)
    List1.AddItem Field.Text
End Sub

```

The CESN transaction is waiting for input from the user, the program could enter text into some fields and continue the transaction, in this example we simply end the transaction by sending PF3 to CICS.

```

    'Send PF3 back to CICS to end CESN
    Screen.SetAID ccIPF3
    Terminal.Send Session
    'Output the text from a 3270 field
    Set Field = Screen.FieldByIndex(1)
    List1.AddItem Field.Text
End Sub

```

Lastly the terminal should be disconnected, and the EPI terminated. ***The order in which this is done is very important.***

```
Sub EPIDone_Click()  
    Term.Disconnect  
    'Delete the EPI COM objects  
    Set Field = Nothing  
    Set Screen = Nothing  
    Set Session = Nothing  
    Set Terminal = Nothing  
    Set EPI = Nothing  
End Sub
```

### EPI call synchronization types

The CICS client EPI COM classes support synchronous (“blocking”) and deferred synchronous (“polling”) protocols. The Visual Basic environment does not support the asynchronous calls that are available in the C++ classes.

In the previous example a Session object was used with the default synchronization type of `cclSync`. When this Session object was used as the first parameter on **Terminal.Start** or **Terminal.Send**, a synchronous link call was made to CICS. The Visual Basic program was then blocked until the reply was received from CICS. When the call returned updated screen data from CICS was immediately available in the Screen object.

To make a deferred synchronous call you use the **Session.SetSyncType** method to set the Session to `cclDSync`. When this Session object is used on a **Terminal.Start** or **Terminal.Send** call, the screen contents are transmitted to CICS as 3270 datastream, but the method returns immediately. This allows the Visual Basic program to continue other tasks, including user interactions, while the CICS server transaction is running. Further 3270 screen updates from CICS must be retrieved later using the **Poll** method on the Terminal object:

```
Sub EPIDSync_Click()  
    'Create a session object (deferred synchronous)  
    Set Session = New CcIOSession  
    Session.SetSyncType cclDSync  
    Terminal.Start Session, "CESN", ""  
End Sub
```

The transaction is now in progress in the CICS server. At a later stage (in response to a user action, or when the Visual Basic program has completed some other task) the **Terminal.PollForReply** method is used to collect the reply from CICS:

```

Sub EPIReply_Click()
    If terminal.State <> cclDiscon And terminal.State <> cclError Then
        If terminal.PollForReply Then
            'Screen has been updated, output some fields
            Set Screen = Terminal.Screen
            Set Field = Screen.FieldByIndex(1)
            List1.AddItem Field.Text
        Else
            List1.AddItem "No Reply from CICS yet"
        End If
    End If
End Sub

```

A CICS server transaction may send more than one reply in response to a **Terminal.Start** or **Terminal.Send** call. More than one **Terminal.PollForReply** call may therefore be needed to collect all the replies. Use the **Terminal.State** method to find out if further replies are expected. If there are, the value returned will be **cclServer**.

## CICS Server Information

The **EPI** COM class provides the names and descriptions of CICS servers configured in the Client initialization file.

An EPI object is created as in the previous examples, and a and a list of CICS server names and their descriptions is output to a listbox “List1”:

```

Sub EPIServers_Click()
    Dim I
    'Instantiate CICS EPI object
    Set EPI = New CcIOEPI
    'List CICS server information
    For I = 1 To EPI.ServerCount
        List1.AddItem EPI.ServerName(I)
        List1.AddItem EPI.ServerDesc(I)
    Next

```

## Using BMS Map data with EPI COM classes

Many CICS server programs use Basic Mapping Support (BMS) to implement their 3270 screen designs. The server programs can then use symbolic names for the individual screen maps and for the 3270 fields on those maps. If the BMS source files are available, they can be copied to the CICS client development environment and used in the implementation of a Visual Basic EPI program.

The CICS BMS Conversion Utility (CICSBMSC.EXE) provided with the CICS client produces a Visual Basic definitions file (a .BAS file) from the source BMS file (.BMS file). This definitions file can then be included in a Visual Basic program, and the same symbolic names used to identify maps and their fields in the server program can be used in the client program with the **EPI Map** COM class.

## Guide to COM CICS EPI

The /B option should be specified when running the conversion utility to produce Visual Basic definitions:

```
CICSBMSC /B <filename>.BMS
```

Running the conversion utility on the BMS file for the supplied sample server program EPIINQ produces the following Visual Basic definitions

```
Public Const MAPINQ = xx.xx.xx.xx.xx.xx.xx
Public Const MAPINQ1_PRODNAME = xx
Public Const MAPINQ1_APPLID = xx
Public Const MAPINQ1_TIME = xx
```

The first constant MAPINQ1 identifies the map and provides information describing the position, size and layout of the map, remaining constants define the named fields within the map.

The following example shows how to use the **Map** COM class to access fields by their BMS symbolic names:

```
Dim EPI As CcIOEPI
Dim Terminal As CcIOTerminal
Dim Session As CcIOSession
Dim Screen As CcIOScreen
Dim Map as CcIOMap
Dim Field As CcIOField
```

First the EPI is initialized and a 3270 terminal connection to CICS is started as in the earlier example:

```
Sub EPIConnect_Click()
    'Create CcIOEPI first to initialize EPI
    Set EPI = New CcIOEPI
    'Create a terminal object and connect to CICS
    Set Terminal = New CcIOTerminal
    Terminal.Connect "CICSNAME", "", ""
    'Create a session object (defaults to synchronous)
    Set Session = New CcIOSession
End Sub
```

Then the BMS application is started. This example uses a transaction code “EPIC” which runs the supplied server program EPIINQ:

```
Sub EPIRunBMS_Click()
    Terminal.Start Session, "EPIC", ""
    Set Screen = Terminal.Screen
```

At this point the CICS server program has returned the first screen to the client. This is expected to be a known map “MAPINQ1” so we create a Map object, and use the **Map.Validate** method to initialize it and to verify that we

received the expected 3270 screen. Fields can then be accessed using the **Map.FieldByName** method:

```
Set Map = New CclOMap
If (Map.Validate(Screen,MAPINQ1)) Then
    Set Field = Map.FieldByName(MAPINQ1_PRODNAM)
    List1.AddItem Field.Text
    Set Field = Map.FieldByName(MAPINQ1_TIME)
    List1.AddItem Field.Text
Else
    List1.Text= "Unexpected screen data"
End If
```

A more complex application would then enter data into selected fields, set the required AID key (Enter, Clear, PF or PA key) and navigate through further screens as required. The client application can mix the use of the **Screen** COM class (and its **FieldByIndex** and **FieldByPosition** methods) with the use of the **Map** COM class.

---

## Connecting to CICS 3270 applications using VBScript

This is again very similar to Visual Basic but differs in how you create the objects. You do not need to have to Dim your variables but it is good coding practice to do so. To create objects you must use the **CreateObject** function, for example:

```
Sub EPICheck_Click()
    ' Create Ccl.EPI first to initialise EPI
    Set EPI = CreateObject("Ccl.EPI")
    ' Create a terminal object and connect to CICS
    Set Terminal = CreateObject("Ccl.Terminal")
    Terminal.Connect "CICSNAME","", ""
    ' Create a session object (defaults to synchronous)
    Set Session = CreateObject("Ccl.Session")
End Sub
```

In a similar manner, to create a Map object you issue

```
Set Map = CreateObject("Ccl.MAP")
```

Screen objects and Fields Objects are created for you.

---

## Support for Automatic Transaction Initiation (ATI)

The CICS server API call EXEC CICS START allows a server program to start a transaction on a particular terminal. This mechanism, called Automatic Transaction Initiation (ATI), requires additional programming at the client side to handle the interaction between these transactions and normal client-initiated transactions.

Client applications can control whether ATI transactions are allowed by using the **setATI** and **queryATI** methods on the Terminal COM class. The default setting is for ATIs to be disabled. The following code fragment shows how to enable ATIs for a particular terminal:

```
// Create terminal connection to CICS server
Dim terminal as CclTerminal
Set terminal = new CclTerminal
terminal.details "MYSERVER","", ""
terminal.setATI CclATIEnabled
```

The CICS client queues ATIs for a terminal while a transaction is in progress. The Ccl Terminal class runs any outstanding ATIs as soon as a transaction ends, and calls Additional programming needed to handle the ATI replies, and to run ATIs before or between client-initiated transactions, depending on the call synchronization type used:

### Synchronous

When you call the Terminal **send** method, any outstanding ATIs are run after the client-initiated transaction has completed. The Terminal class waits for the ATI replies then updates the CclOScreen object contents as part of the synchronous **send** call. If you expect an ATI to occur before or between client-initiated transactions, call the Ccl Terminal **receiveATI** method to wait synchronously for the ATI.

### Deferred synchronous

After the CclTerminal **Start** or **Send** method is called for a deferred synchronous session, the **Poll** or **PollForReply** method is used to receive the replies. Outstanding ATIs are started when the last reply is received (that is on the final **Poll** or **PollForReply** method). You can also call the **Poll** or **PollForReply** method to start and receive replies for ATIs between client-initiated transactions.

As the **Poll** or **PollForReply** methods can be called before or between client-initiated transactions, the **receiveATI** method is not needed (and is invalid) for deferred synchronous sessions.

---

## Security Management

You are now able to do security management on Servers that support Password Expiry Management. Please refer to the *CICS Family: Client/Server Programming* for more information on supported servers and protocols.

To use these features you first must have created a Terminal object and invoked the **SetTerminalDefinition** method to associate a userid and password with the object. The two methods available are **VerifyPassword** which checks the userid and password within the terminal object with the

Server Security System, and **ChangePassword** which allows you to change the password at the server. If successful, the terminal object password is updated accordingly.

If either call is successful, you are returned a CclOSecAttr object. This object provides access to information such as last verified Date and Time, Expiry Date and Time and Last access Date and Time. If you query for example last verified Date, you are returned a CclOSecTime object which allows you to get the information in various formats. The following shows the use of these various objects.

```
' Terminal object already created called term

on error goto pemhandler

dim SecAttr as CclOSecAttr
dim LastVerified as CclOSecTime
dim lvddate as Date

set SecAttr = term.VerifyPassword
set LastVerified = SecAttr.LastVerifiedTime

lvddate = LastVerified.GetDate
strout = Format(lvddate, "hh:mm:ss, dddd, mmm d yyyy")
Text1.Text = strout

exit sub

pemhandler:
' handle a expired password here maybe

end sub
```





## Part 2. COM classes — Reference

<b>Chapter 4. Buffer COM class</b> . . . . .	33	ExCode	44
Interface Selection . . . . .	33	ExCodeText	44
Object Creation . . . . .	33	ServerCount	45
Methods . . . . .	34	ServerDesc	45
AppendString . . . . .	34	ServerName	45
Data . . . . .	34	SetErrorFormat	45
ExtractString . . . . .	34		
InsertString . . . . .	34	<b>Chapter 7. EPI COM class</b> . . . . .	47
Length . . . . .	34	Interface Selection . . . . .	47
Overlay . . . . .	35	Object Creation . . . . .	47
SetData . . . . .	35	Methods . . . . .	47
SetLength . . . . .	35	Diagnose . . . . .	47
SetString . . . . .	35	ErrorFormat . . . . .	47
String . . . . .	35	ErrorOffset . . . . .	48
		ErrorWindow . . . . .	48
<b>Chapter 5. Connect COM class</b> . . . . .	37	ExCode . . . . .	48
Interface Selection . . . . .	37	ExCodeText . . . . .	49
Object Creation . . . . .	37	ServerCount . . . . .	49
Methods . . . . .	37	ServerDesc . . . . .	49
AlterSecurity . . . . .	37	ServerName . . . . .	49
Cancel . . . . .	38	SetErrorFormat . . . . .	49
Changed . . . . .	38	State . . . . .	50
ChangePassword . . . . .	38	Terminate . . . . .	50
Details . . . . .	39		
Link . . . . .	39	<b>Chapter 8. Field COM class</b> . . . . .	51
MakeSecurityDefault . . . . .	40	Interface Selection . . . . .	51
Password . . . . .	40	Methods . . . . .	51
ServerName . . . . .	40	AppendText . . . . .	51
ServerStatus . . . . .	40	BackgroundColor . . . . .	51
ServerStatusText . . . . .	41	BaseAttribute . . . . .	52
Status . . . . .	41	Column . . . . .	52
TranDetails . . . . .	41	DataTag . . . . .	52
UnpaddedPassword . . . . .	41	ForegroundColor . . . . .	52
UnpaddedServerName . . . . .	42	Highlight . . . . .	53
UnpaddedUserid . . . . .	42	InputProt . . . . .	53
UserId . . . . .	42	InputType . . . . .	53
VerifyPassword . . . . .	42	Intensity . . . . .	54
		Length . . . . .	54
<b>Chapter 6. ECI COM class</b> . . . . .	43	Position . . . . .	54
Interface Selection . . . . .	43	ResetDataTag . . . . .	54
Object Creation . . . . .	43	Row . . . . .	54
Methods . . . . .	43	SetBaseAttribute . . . . .	54
ErrorFormat . . . . .	43	SetExtAttribute . . . . .	55
ErrorOffset . . . . .	43	SetText . . . . .	55
ErrorWindow . . . . .	44	Text . . . . .	55

TextLength . . . . .	55	<b>Chapter 13. SecTime COM class . . . . .</b>	69
Transparency. . . . .	55	Interface Selection . . . . .	69
<b>Chapter 9. Flow COM class . . . . .</b>	57	Public Methods . . . . .	69
Interface Selection . . . . .	57	Day . . . . .	69
Object Creation . . . . .	57	GetDate . . . . .	69
Methods . . . . .	57	Hours . . . . .	69
AbendCode . . . . .	57	Hundredths . . . . .	69
CallType . . . . .	57	Minutes . . . . .	70
CallTypeText . . . . .	58	Month . . . . .	70
Diagnose . . . . .	58	Seconds . . . . .	70
Flowid . . . . .	58	Year. . . . .	70
ForceReset . . . . .	58	<b>Chapter 14. Session COM class. . . . .</b>	71
Poll . . . . .	58	Interface Selection . . . . .	71
SetSyncType . . . . .	59	Object Creation . . . . .	71
SetTimeout . . . . .	59	Methods . . . . .	71
SyncType . . . . .	59	Diagnose . . . . .	71
Timeout . . . . .	59	SetSyncType . . . . .	71
Wait. . . . .	59	State . . . . .	72
<b>Chapter 10. Map COM class . . . . .</b>	61	TransId . . . . .	72
Interface Selection . . . . .	61	<b>Chapter 15. Terminal COM class . . . . .</b>	73
Object Creation . . . . .	61	Interface Selection . . . . .	73
Methods . . . . .	61	Object Creation . . . . .	73
ExCode . . . . .	61	Methods . . . . .	73
FieldByName . . . . .	62	AlterSecurity. . . . .	73
Validate . . . . .	62	CCSID . . . . .	74
<b>Chapter 11. Screen COM class . . . . .</b>	63	ChangePassword . . . . .	74
Interface Selection . . . . .	63	Connect . . . . .	74
Methods . . . . .	63	Devtype . . . . .	75
CursorCol. . . . .	63	Diagnose . . . . .	75
CursorRow . . . . .	63	Disconnect . . . . .	75
Depth . . . . .	63	DisconnectWithPurge . . . . .	75
FieldByIndex. . . . .	64	DiscReason . . . . .	75
FieldByPosition . . . . .	64	ExCode . . . . .	75
FieldCount . . . . .	64	ExCodeText . . . . .	76
MapName . . . . .	64	Install . . . . .	76
MapSetName . . . . .	64	MakeSecurityDefault . . . . .	76
SetAID. . . . .	65	NetName . . . . .	76
SetCursor . . . . .	65	Password . . . . .	77
Width . . . . .	65	Poll . . . . .	77
<b>Chapter 12. SecAttr COM class . . . . .</b>	67	PollForReply. . . . .	77
Interface Selection . . . . .	67	QueryATI. . . . .	77
Public Methods . . . . .	67	ReadTimeout. . . . .	78
ExpiryTime . . . . .	67	ReceiveATI . . . . .	78
InvalidCount. . . . .	67	Screen . . . . .	78
LastAccessTime . . . . .	68	Send . . . . .	78
LastVerifiedTime . . . . .	68	ServerName . . . . .	78
		SetATI . . . . .	79
		SetTermDefns . . . . .	79

SignonCapability . . . . .	80	Interface Selection . . . . .	83
Start . . . . .	80	Object Creation . . . . .	83
State . . . . .	81	Methods . . . . .	83
TermId. . . . .	81	BackOut . . . . .	83
TransId . . . . .	81	Commit . . . . .	84
Userid . . . . .	81	ForceReset . . . . .	84
VerifyPassword . . . . .	81	UowId . . . . .	84

**Chapter 16. UOW COM class. . . . . 83**

The following chapters contain descriptions of all the Client COM classes, in alphabetic order. Within the interfaces, there are alphabetical lists of methods. For further information on how to use these interfaces, see Part 1.

You can find summaries of these methods using the Visual Basic Object Browser. ECI classes and methods are listed in the CcIECILib library and EPI classes are listed in the CcIEPILib library. Ensure you have added the appropriate reference to your Visual Basic Project first.



---

## Chapter 4. Buffer COM class

A CcIOBuffer object contains a data area in memory which can be used to hold information. A particular use for a CcIOBuffer object is to hold a COMMAREA used to pass data to and from a CICS server.

The CcIOBuffer object is primarily intended for use with byte (binary) data. Typically a COMMAREA will contain an application-specific data structure, often originating from a CICS server COBOL, PL/1 or C program. The preferred method for handling binary data in Visual Basic is now the Byte data type. The **SetData** and **Data** methods allow the contents of the CcIOBuffer object to be accessed as a Byte array. The CcIOBuffer object can be used for string data and it will store strings as single-byte ANSI characters, but it does not provide any support for code-page conversions or DBCS. Note that in 32-bit environments Visual Basic uses 2-byte Unicode character representation, the COM class converts this to and from single-byte ANSI.

When a CcIOBuffer object is created it allocates an area of memory as its buffer. The length of this buffer can be set explicitly via the **SetLength** method.

---

### Interface Selection

For visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as CcIOBuf
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Object Creation

You can create an object in 2 ways

```
set var = CreateObject("Ccl.Buffer")  
set var = New CcIOBuf
```

The preferred method in Visual Basic is the use of **New**. For VBScript, you can only use the **CreateObject** method.

---

### Methods

#### AppendString

**AppendString**(*string* as String)

*string*

The source string.

Appends a string to existing data in the **Ccl.Buffer** object.

#### Data

**Data()** as Variant

Returns the contents of the buffer as a Byte array.

#### ExtractString

**ExtractString** (*offset* as Integer[,  
                  *length* as Integer]) as String

*offset*

The offset into the data area.

*length*

(optional) The length, in bytes, of the string to be extracted.

Returns a string from the data area starting at the specified offset.

If *length* is not specified, **ExtractString** will return data until it finds the first null terminator. If *length* is specified, **ExtractString** will return the number of bytes requested, including any nulls found in the string.

#### InsertString

**InsertString** (*offset* as Integer,  
                  *string* as String)

*offset*

The offset in the data area where the string is to be inserted.

*string*

The source string.

Inserts the given string into the data area at the given offset.

#### Length

**Length()** as Integer

Returns the length of the data area in bytes.

## Overlay

**Overlay** (*offset* as Integer,  
*string* as String)

*offset*

The offset in the data area where the string is to be inserted.

*string*

The source string.

Overlays the data area with the given string, starting at the given offset.

## SetData

**SetData**(*array* as Variant)

*array*

The array containing the source data.

Copies the supplied array into the buffer. Byte, Integer, and Long arrays are supported.

## SetLength

**SetLength**(*length* as Integer)

*length*

The new length of the data area, in bytes

Changes the current length of the data area. If you increase the length of the buffer object, the extra space is padded with nulls. The client will truncate any nulls before sending the buffer to a CICS server.

## SetString

**SetString**(*string* as String)

*string*

Source string

Copies the supplied string into the object.

## String

**String()** as String

Returns, as a string, the contents of the **Ccl.Buffer** object.





---

## Chapter 5. Connect COM class

The **Connect** COM class is used to maintain and represent an ECI connection between a client and a named server. Access to the server is optionally controlled by a user ID and password. It can call a program in the server or get information on the state of the connection.

Before the **Connect** COM class can be used to make calls to CICS, it should be initialized using the **Details** method and, optionally, the **TranDetails** method.

Any interaction between client and server requires a **CclOFlow** object and a **CclOConnect** object.

---

### Interface Selection

For visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as CclOConn
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Object Creation

You can create an object in 2 ways

```
set var = CreateObject("Ccl.Connect")  
set var = New CclOConn
```

The preferred method in Visual Basic is the use of **New**. For VBScript, you can only use the **CreateObject** method.

---

### Methods

#### AlterSecurity

**AlterSecurity**(*newUserId* as String, *newPassword* as String)

*newUserId*

The new userid

*newPassword*

The new password corresponding to the new userid.

## COM class: Connect

Sets the userid and password to be used on the next link call.

### Cancel

**Cancel**(*flow* as Object)

or

**Cancel**(*flow* as CclOFlow)

*flow*

The CclOFlow object used to control the client/server call  
Cancels any **Changed** call which was previously issued to the server associated with this connection.

### Changed

**Changed**(*flow* as Object)

or

**Changed**(*flow* as CclOFlow)

*flow*

The CclOFlow object used to control the client/server call  
Requests the server to notify the client when the current connection status changes. The call is ignored if there is an outstanding **Changed** call for this connection.

### ChangePassword

**ChangePassword** (*newPassword* as String) as Object

or

**ChangePassword** (*newPassword* as String) as CclOSecAttr

*newPassword*

The new password  
Allows a client application to change the password held in the Connect object and the password recorded by an external security manager for the userid held in the Connect object. The external security manager is assumed to be located in the server defined by the Connect object. A CclOSecAttr object is returned if no errors occur.

## Details

**Details** (*serverName* as String,  
*userId* as String,  
*password* as String)

### *serverName*

The name of the server. If no name is supplied the default server—the first server named in the Client initialization file—is used. You can discover this name, after the first call to the server by using the **ServerName** method. The length is adjusted to 8 characters by padding with blanks.

### *userId*

The user ID, if needed. The length is adjusted to 16 characters by padding with blanks.

### *password*

The password corresponding to the user ID in *userId*, if needed. The length is adjusted to 16 characters by padding with blanks.

Use this method to supply details of the CICS server. No interaction with the CICS server takes place until the **Link**, **Status** or **Changed** methods are called. The user ID and password are not needed if the connection is only used for status calls or if the server has no security.

## Link

**Link** (*flow* as Object,  
*programName* as String,  
*commArea* as Object,  
*unitOfWork* as Object)

or

**Link** (*flow* as CclOFlow,  
*programName* as String,  
*commArea* as CclOBuf,  
*unitOfWork* as CclOUOW)

### *flow*

The CclOFlow object used to control the client/server call.

### *programName*

The name of the server program that is being called. The length is adjusted to 8 characters by padding with blanks or truncating, if necessary.

## COM class: Connect

### *commArea*

A CclOBuffer object that holds the data to be passed to the called program in a COMMAREA. A NULL value should be supplied if no COMMAREA is to be sent.

### *unitOfWork*

The CclOUOW object that identifies the unit of work (UOW) with which this call is being associated. A NULL value should be supplied if no UOW is to be used.

Calls the specified program on the server. The server program sees the incoming call as an EXEC CICS LINK call.

## MakeSecurityDefault

### **MakeSecurityDefault()**

Informs the client that the current userid and password for this object is to become the default for ECI and EPI requests passed to the server as specified in the construction of the Connect object.

## Password

### **Password() as String**

Returns the password held by the CclOConnect object, padded with spaces.

## ServerName

### **ServerName() as String**

Returns the name of the server system held by the CclOConnect object and listed by the Client initialization file, or blanks if the default server is being used and no calls have yet been made.

## ServerStatus

### **ServerStatus() as Integer**

or

### **ServerStatus() as CclConnectStatusCodes**

Returns the status of the server connection, set by an earlier **status** or **changed** request. Possible values are:

#### **cclUnknown**

The CICS server status is unknown

#### **cclAvailable**

The CICS server is available

#### **cclUnavailable**

The CICS server is not available

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

## ServerStatusText

**ServerStatusText()** as String

Returns a string, set by an earlier **status** or **changed** request, indicating the availability of the server.

## Status

**Status**(*flow* as Object)

or

**Status**(*flow* as CcIOFlow)

*flow*

The CcIOFlow object used to control the client/server call  
Request the status of the server connection.

## TranDetails

**TranDetails** (*runTran* as String,  
*attachTran* as String)

*runTran*

The CICS transaction under which called programs will run. The default is to use the default server transaction. The length is adjusted to 4 characters by padding with blanks.

*attachTran*

The CICS transaction to which called programs are attached. The default is to use the default CPML. The length is adjusted to 4 characters by padding with blanks.

This method is used to supply additional information to the CICS server. The information is optional, but can be used to affect the environment in which programs are run on the CICS server.

## UnpaddedPassword

**UnpaddedPassword()** as String

Returns the password held by the CcIOConnect object, but with no padding with spaces at the end.

## COM class: Connect

### UnpaddedServerName

**UnpaddedServerName() as String**

Returns the server name held by the CclOConnect object, but with no padding with spaces at the end.

### UnpaddedUserid

**UnpaddedUserid() as String**

Returns the userid held by the CclOConnect object, but with no padding with spaces at the end.

### UserId

**UserId() as String**

Returns the user ID held by the CclOConnect object, padded with spaces, or blanks if none.

### VerifyPassword

**VerifyPassword() as Object**

or

**VerifyPassword() as CclOSecAttr**

Allows a client application to verify that the password held in the Connect object matches the password recorded by an external security manager for the userid held in the Connect object. The external security manager is assumed to be located in the server defined by the Connect object. A CclOSecAttr Object is returned if no errors occur.

---

## Chapter 6. ECI COM class

All applications using the ECI COM class must first create a CcIOECI object.

The ECI COM class provides details of candidate CICS servers. It can also be used to obtain error information.

---

### Interface Selection

For visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as CcIOECI
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Object Creation

You can create an object in 2 ways

```
set var = CreateObject("Ccl.ECI")  
set var = New CcIOECI
```

The preferred method in Visual Basic is the use of **New**. For VBScript, you can only use the **CreateObject** method.

---

### Methods

#### ErrorFormat

##### **ErrorFormat() as Integer**

Returns a value indicating the current setting for the Error Message Format. Refer to SetErrorFormat for a current list of valid values.

#### ErrorOffset

##### **ErrorOffset() as Long**

Returns a value which can be used to convert a CICS Client error value retrieved from the ERR.Number method into the documented ExCode error values. For more information on how to do this, refer to "Exception Handling" on page 12.

## ErrorWindow

### Deprecated method

Do not use this method in new applications. The method has been deprecated and is only provided for backwards compatibility.

**ErrorWindow**(*display* as Boolean)

*display*

- true** Permits the error window to be displayed to the user. This is the default setting.
- false** The error window will not be displayed to the user. The application must check for errors using the **ExCode** method.

## ExCode

### Deprecated method

Do not use this method in new applications. The method has been deprecated and is only provided for backwards compatibility.

**ExCode**() as Integer

or

**ExCode**() as CclECIExceptionCodes

Returns an enumeration that indicates the last ECI error.

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

Note that the **ExCodeText** method returns a descriptive text string for the error value.

## ExCodeText

### Deprecated method

Do not use this method in new applications. The method has been deprecated and is only provided for backwards compatibility.

**ExCodeText**() as String



Returns a string containing descriptive text for the last ECI error.

## ServerCount

**ServerCount()** as Integer

Returns the number of candidate servers to which the client may be connected, as configured in the Client initialization file.

## ServerDesc

**ServerDesc(*index* as Integer)** as String

*index*

The number of a connected server in the list, starting from 1

Returns the description of the *index*th server.

## ServerName

**ServerName(*index* as Integer)** as String

*index*

The number of a connected server in the list, starting from 1

Returns the name of the *index*th server.

## SetErrorFormat

**SetErrorFormat(*format* as Integer)**

*format*

**0** Old format, provided for backward compatability only.

**1** New format, provides more information in the Visual Basic and VBScript **Err** object. This is the recommended format to use.

This method allows you to select an error message format.



---

## Chapter 7. EPI COM class

The EPI COM class initializes the CICS client EPI function. It also has methods that allow you to obtain information about CICS servers which could be used. You create a CclOEPI object before you create CclOTerminal objects to connect to CICS servers. The **Diagnose**, **ExCode**, and **State** methods provide information on error conditions.

---

### Interface Selection

For visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as CclOEPI
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Object Creation

You can create an object in 2 ways

```
set var = CreateObject("Ccl.EPI")  
set var = New CclOEPI
```

The preferred method in Visual Basic is the use of **New**. For VBScript, you can only use the **CreateObject** method.

---

### Methods

#### Diagnose

**Diagnose()** as String

Returns a character string which holds a description of the last error.

#### ErrorFormat

**ErrorFormat()** as Integer

Returns a value indicating the current setting for the Error Message Format. Refer to "SetErrorFormat" on page 49 for a current list of valid values.

## ErrorOffset

### ErrorOffset() as Long

Returns a value which can be used to convert a CICS Client error value retrieved from the ERR.Number method into the documented ExCode error values. For more information on how to do this, refer to “Exception Handling” on page 12.

## ErrorWindow

### Deprecated method

Do not use this method in new applications. The method has been deprecated and is only provided for backwards compatibility.

### ErrorWindow(*display* as Boolean)

#### *display*

- true** Permits the error window to be displayed to the user. This is the default setting.
- false** The error window will not be displayed to the user. The application must check for errors using the **ExCode** method.

## ExCode

### Deprecated method

Do not use this method in new applications. The method has been deprecated and is only provided for backwards compatibility.

### ExCode() as Integer

or

### ExCode() as CclEPIExceptionCodes

Returns the condition code. Possible values are:

#### **cclSystemError**

An internal CICS client system error occurred.

#### **cclUnknownServer**

There is no CICS server corresponding to the supplied *index* on **ServerDesc** or **ServerName** methods.

#### **cclNoError**

The call has executed normally.

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

## ExCodeText

### Deprecated method

Do not use this method in new applications. The method has been deprecated and is only provided for backwards compatibility.

### **ExCodeText() as String**

Returns a string containing descriptive text for the most recent exception.

## ServerCount

### **ServerCount() as Integer**

Returns the number of candidate servers to which the client may be connected, as configured in the Client initialization file.

## ServerDesc

### **ServerDesc(*index* as Integer) as String**

*index*

The index number of a connected server (starting from 1).

Returns a description of the selected CICS server, or a NULL string if no information is available in the Client initialization file for the specified server.

## ServerName

### **ServerName(*index* as Integer) as String**

*index*

The index number of a connected server (starting from 1).

Returns the name of the requested CICS server, or a NULL string if no information is available in the Client initialization file for the specified server.

## SetErrorFormat

### **SetErrorFormat(*format* as Integer)**

*format*

**0** Old format, provided for backward compatibility only.

**1** New format, provides more information in the Visual Basic and VBScript **Err** object. This is the recommended format to use.

This method allows you to select an error message format.

## State

**State()** as Integer

or

**State()** as CclEPIStates

Returns a value which indicates the state of the EPI. Possible values are:

**cclActive**

Initialized

**cclDiscon**

Terminated

**cclError**

Error, refer to “Exception Handling” on page 12.

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

## Terminate

### Deprecated method

Do not use this method in new applications. The method has been deprecated and is only provided for backwards compatibility.

**Terminate()**

Terminates the CICS client EPI in a controlled manner.

---

## Chapter 8. Field COM class

The **Field** COM class is used to access a single field on a 3270 screen.

CclOField objects are created and deleted when 3270 data from the CICS server is processed by a CclOScreen object.

Field objects are returned by invoking a CclOScreen object's **fieldbyIndex** or **fieldbyPosition** method. For example:

```
set var=Screen.fieldbyIndex(1)
```

Methods in this class allow field text and attributes to be read and updated. Updated fields are sent to the CICS server on the next transmission.

---

### Interface Selection

For visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as CclOField
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Methods

#### AppendText

**AppendText(textString as String)**

*textString*

The text string to be appended to the field  
Appends the characters within *textString* to the end of the text already in the field.

#### BackgroundColor

**BackgroundColor() as Integer**

or

**BackgroundColor() as CclColorAttributes**

## COM class: Field

Returns a value which indicates the background color of the field as listed in “CclField Color Attributes” on page 95.

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

### BaseAttribute

#### **BaseAttribute() as Integer**

Returns the 3270 base attribute of the field.

### Column

#### **Column() as Integer**

Returns the column number of the position of the start of the field on the screen, with the leftmost column being 1.

### DataTag

#### **DataTag() as Integer**

or

#### **DataTag() as CclModifiedAttributes**

Returns a value which indicates whether the data in the field has been modified. Possible values are:

cclModified  
cclUnmodified

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

### ForegroundColor

#### **ForegroundColor() as Integer**

or

#### **ForegroundColor() as CclColorAttributes**

Returns a value which indicates the foreground color of the field as listed in “CclField Color Attributes” on page 95.

Constants are available in the type library. Use the Visual Basic Object Browser to view them.



## Highlight

**Highlight() as Integer**

or

**Highlight() as CclHighlightAttributes**

Returns a value which indicates which type of highlight is being used as listed in “CclField Highlight Attributes” on page 94.

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

## InputProt

**InputProt() as Integer**

or

**InputProt() as CclProtAttributes**

Returns a value which indicates whether the field is protected. Possible values are:

- cclProtect
- cclUnprotect

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

## InputType

**InputType() as Integer**

or

**InputType() as CclNumericAttributes**

Returns a value which indicates whether the field is alphanumeric or numeric. Possible values are:

- cclAlphanumeric
- cclNumeric

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

## COM class: Field

### Intensity

**Intensity() as Integer**

or

**Intensity() as CclIntensityAttributes**

Returns a value which indicates whether the field is normal, intense or dark.

Possible values are:

cclDark

cclNormal

cclIntense

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

### Length

**Length() as Integer**

Returns the total length of the field. This includes one byte used to store the 3270 attribute byte information therefore the actual space for data is one less than the value returned by this method. See also the TextLength method.

### Position

**Position() as Integer**

Returns the position of the start of the field as an offset from the top left corner of the screen. The top row consists of positions 0 to 79; the second row, positions 80 to 159; etc.

### ResetDataTag

**ResetDataTag()**

Resets the modified data tag (MDT) to cclUnmodified.

### Row

**Row() as Integer**

Returns the row number of the position of the start of the field on the screen. The top row is 1.

### SetBaseAttribute

**SetBaseAttribute(*Attribute* as Integer)**

*Attribute*

The value of the base 3270 attribute to be entered into the field.  
Sets the 3270 base attribute.

**SetExtAttribute**

**SetExtAttribute**(*Attribute* as Integer, *Value* as Integer)

*Attribute*

The type of extended attribute to be set

**Value**

The value of the extended attribute  
Sets the extended 3270 attribute. If an invalid 3270 attribute type or value is supplied a parameter exception is raised.

**SetText**

**SetText**(*textString* as String)

*textString*

The null-terminated text to be entered into the field  
Copies *textString* into the field.

**Text**

**Text**() as String

Returns the text currently held in the field.

**TextLength**

**TextLength**() as Integer

Returns the number of characters currently held in the field.

**Transparency**

**Transparency**() as Integer

or

**Transparency**() as CclTransparencyAttributes

Returns a value which indicates the background transparency of the field as listed in "CclField Transparency Attributes" on page 95.

Constants are available in the type library. Use the Visual Basic Object Browser to view them.



---

## Chapter 9. Flow COM class

A CclOFlow object is used to control ECI communications for a client/server pair.

A CclOFlow object is created for each client server interaction (call from client and response from server) and destroyed when it has been used. CclOFlow objects can be reused but an attempt to reuse a CclOFlow object that is already in use is rejected.

---

### Interface Selection

For visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as CclOFlow
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Object Creation

You can create an object in 2 ways

```
set var = CreateObject("Ccl.Flow")  
set var = New CclOFlow
```

The preferred method in Visual Basic is the use of **New**. For VBScript, you can only use the **CreateObject** method.

---

### Methods

#### AbendCode

**AbendCode() as String**

Returns a four-character CICS transaction abend code or spaces if no abend has occurred.

#### CallType

**CallType() as Integer**

or

## COM class: Flow

### **CallType() as CclFlowCallTypes**

Returns the type of call the flow is currently executing.current

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

### **CallTypeText**

#### **CallTypeText() as String**

Returns the type of call the flow is currently executing in text format.

### **Diagnose**

#### **Diagnose() as String**

Returns text describing the current state of the flow object.

### **Flowid**

#### **Flowid() as Integer**

Returns a unique identifier for this flow object.

### **ForceReset**

#### **ForceReset()**

Makes the flow inactive and resets it. Typically, this method is used to prepare a flow object for re-use or deletion after a flow has been abandoned.

### **Poll**

#### **Poll(commArea as Object) as Boolean**

or

#### **Poll(commArea as CclOBuf) as Boolean**

*commArea*

A CclOBuffer object into which the returned COMMAREA will be placed.

This parameter can be set to **Nothing** if no COMMAREA should be returned.

Indicates whether a reply has been received from a deferred synchronous **Backout**, **Cancel**, **Changed**, **Commit**, **Link**, or **Status** call request. This method is only valid for deferred synchronous communications. Possible values are:

**True** A reply has been received.

**False** A reply has not been received.

**SetSyncType****SetSyncType(syncType as Integer)**

or

**SetSyncType(syncType as CclFlowSyncTypes)***syncType*

The synchronization type required for this CclOFlow object. Possible values are:

cclSync

cclDSync

Sets the synchronization type required for this CclOFlow object. If cclSync is used, **link** and **status** calls using this flow will block the calling program until a reply is received from CICS. If cclDSync is used, **link** and **status** calls using this flow will return immediately to the calling program. The program can then use the **Poll** method to receive the reply from CICS at a later time.

**SetTimeout****SetTimeout(Timeout as Integer)**

Sets the timeout value for the flow object for the next activation of the flow. This value can be set while a flow is active but it does not affect the current active flow.

**SyncType****SyncType() as Integer**

or

**SyncType() as CclFlowSyncTypes**

Returns the type of synchronisation being used.

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

**Timeout****Timeout() as Integer**

Returns the current timeout value set for the flow object.

**Wait****Wait()**

## COM class: Flow

Waits for a reply from the server, blocking the client process in the meantime. This method is used when a deferred synchronous call was made, but the application now wants to wait synchronously for a reply.



---

## Chapter 10. Map COM class

The **Map** COM class provides validation and access to 3270 screen data using symbolic information obtained from CICS BMS maps. To use this interface you will need to run the CICSBMSC utility on your server program BMS maps.

---

### Interface Selection

For visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as Ccl0Map
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Object Creation

You can create an object in 2 ways

```
set var = CreateObject("Ccl.Map")  
set var = New Ccl0Map
```

The preferred method in Visual Basic is the use of **New**. For VBScript, you can only use the **CreateObject** method.

---

### Methods

#### ExCode

##### Deprecated method

Do not use this method in new applications. The method has been deprecated and is only provided for backwards compatibility.

**ExCode() as Integer**

or

**ExCode() as CclEPIExceptionCodes**

Returns a value which indicates the current condition code.

## COM class: Map

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

### FieldByName

**FieldByName**(*name* as Integer) as Object

or

**FieldByName**(*name* as Integer) as CclOField

*name*

Symbolic value for the required field. This value is provided in the <mapname>.BAS file generated from the source BMS by the CICSBMSC utility.

Returns the specified CclOField object.

### Validate

**Validate** (*screenRef* as Object, *mapname* as String) as Boolean

or

**Validate** (*screenRef* as CclOScreen, *mapname* as String) as Boolean

*screenRef*

CclOScreen object

*mapname*

String value supplied in <mapname>.BAS file generated from the source BMS by the CICSBMSC utility.

Validate map against the current screen.

This method can be used to verify that a specific BMS map has been received from the CICS server. Possible return values are:

**TRUE**

Specified BMS map matches current screen contents.

**FALSE**

Specified BMS map does not match current screen contents

If TRUE is returned, the **FieldByName** method can be used to access fields using their BMS name.

---

## Chapter 11. Screen COM class

The **Screen** COM class maintains all data on the 3270 virtual screen and provides access to this data. It contains a collection of CclOField objects which represent the fields on the current 3270 screen.

A single Screen object is created by the Terminal object when the terminal is installed either with the Ccl Terminal **connect** or **install** method. The application gets access to the CclOScreen object via the Ccl Terminal **Screen** method.

---

### Interface Selection

For visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as CclOScreen
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Methods

#### CursorCol

**CursorCol()** as Integer

Returns the current cursor column (leftmost col = 1).

#### CursorRow

**CursorRow()** as Integer

Returns the current cursor row (topmost col = 1).

#### Depth

**Depth()** as Integer

Returns the number of rows on the screen.

## COM class: Screen

### FieldByIndex

**FieldByIndex(*index* as Integer) as Object**

or

**FieldByIndex(*index* as Integer) as CclOField**

*index*

The index number of the field required. The first field is number 1.

### FieldByPosition

**FieldByPosition (*rowPos* as Integer, *colPos* as Integer) as Object**

or

**FieldByPosition (*rowPos* as Integer, *colPos* as Integer) as CclOField**

*rowPos*

The row number of the field (topmost row = 1).

*colPos*

The column number of the field (leftmost column = 1).

### FieldCount

**FieldCount() as Integer**

Returns the number of fields on the screen.

### MapName

**MapName() as String**

Returns a string specifying the name of the map that was most recently referenced in the MAP option of a SEND MAP command processed for the terminal resource. If the terminal resource is not supported by BMS, or the server has no record of any map being sent, the value returned is blank.

### MapSetName

**MapSetName() as String**

Returns a string specifying the name of the mapset that was most recently referenced in the MAPSET option of a SEND MAP command processed for the terminal resource. If the MAPSET option was not specified on the most recent request, BMS used the map name as the mapset name. In both cases, the mapset name used may have been suffixed by a terminal suffix. If the terminal resource is not supported by BMS, or the server has no record of any mapset being sent, the value returned is blank.

**SetAID****SetAID**(*key* as Integer)

or

**SetAID**(*key* as CclADIKeys)*key*

The AID key value as listed in “CclScreen AID key codes” on page 93.

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

Sets the AID key value to be passed to the server on the next transmission.

**SetCursor****SetCursor** (*rowPos* as Integer, *colPos* as Integer)*rowPos*

The required row number of the cursor (topmost row = 1).

*colPos*

The required column number of the cursor (leftmost column = 1).

**Width****Width()** as Integer

Returns the number of columns on the screen.



---

## Chapter 12. SecAttr COM class

The **SecAttr** COM class provides information about passwords reported back by the external security manager when issuing `verifySecurity` or `changePassword` methods on `CcIOConnect` or `CcIOTerminal` objects.

This object is created and owned by the `CcIOConnect` or `CcIOTerminal` Object and access to this object is provided when invoking the **VerifyPassword** or **ChangePassword** methods.

---

### Interface Selection

For visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as CcIOSecAttr
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Public Methods

#### ExpiryTime

**ExpiryTime()** as **Object**

or

**ExpiryTime()** as **CcIOSecTime**

Returns a `CcIOSecTime` object which contains the Date and Time at which the password will expire.

#### InvalidCount

**InvalidCount()** as **Integer**

Returns the number of times that an invalid password has been entered for the userid.

## **LastAccessTime**

**LastAccessTime() as Object**

or

**LastAccessTime() as CclOSecTime**

Returns a CclOSecTime object which contains the date and time when the userid was last accessed.

## **LastVerifiedTime**

**LastVerifiedTime() as Object**

or

**LastVerifiedTime() as CclOSecTime**

Returns a CclOSecTime object which contains the date and time of the last verification.



---

## Chapter 13. SecTime COM class

The **SecTime** COM class provides date and time information in the **CcLOSecAttr** object for various entries reported back by the external security manager when issuing **verifySecurity** or **changePassword** methods on **Connect** or **Terminal** objects.

These objects are created and owned by the **CcLOSecAttr** object and access is obtained via the various methods available on this object. No constructors or destructors are available.

---

### Interface Selection

For visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as CcLOSecTime
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Public Methods

#### Day

**unsigned short Day() as Integer**  
Returns the day in the range 1 to 31.

#### GetDate

**GetDate() as Date**  
Returns the date and time in a Visual Basic DATE type format.

#### Hours

**unsigned short Hours() as Integer**  
Returns the hours in the range 0 to 23.

#### Hundredths

**unsigned short Hundredths() as Integer**  
Returns the hundredths of a second in the range 0 to 99.

## **Minutes**

**unsigned short Minutes() as Integer**

Returns the minutes in the range 0 to 59.

## **Month**

**unsigned short Month() as Integer**

Returns the month in the range 1 to 12.

## **Seconds**

**unsigned short Seconds() as Integer**

Returns the seconds in the range 0 to 59.

## **Year**

**unsigned short Year() as Integer**

Returns a 4 digit year.

---

## Chapter 14. Session COM class

The **Session** COM class controls the flow of data to and from CICS within a single EPI session.

---

### Interface Selection

For visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as Ccl0Session
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Object Creation

You can create an object in 2 ways

```
set var = CreateObject("Ccl.Session")  
set var = New Ccl0Session
```

The preferred method in Visual Basic is the use of **New**. For VBScript, you can only use the **CreateObject** method.

---

### Methods

#### Diagnose

##### **Diagnose()** as String

Returns the text description of the current state of the session.

#### SetSyncType

##### **SetSyncType(syncType as Integer)**

or

##### **SetSyncType(syncType as CclFlowSyncTypes)**

*syncType*

The synchronization type required for this Ccl0Session object. Possible values are:

cclSync

## COM class: Session

### cclDSync

Sets the synchronization type required for this CclOSession object. If cclSync is used, **Start** and **Send** calls using this flow will block the calling program until a reply is received from CICS. If cclDSync is used, **Start** and **Send** calls using this flow will return immediately to the calling program. The program can then use the **Poll** method to receive the reply from CICS at a later time.

## State

### State() as Integer

or

### State() as CclEPISStates

Returns a value which indicates the current state of the session. Possible values are:

#### cclActive

Connected

#### cclServer

Transaction in progress in the CICS server.

#### cclClient

CICS server is waiting for a response from the client

#### cclDiscon

Disconnected

#### cclError

Error, call **ExCode** and **Diagnose** methods for further information.

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

## TransId

### TransId() as String

Returns the four-letter name of the current transaction.

---

## Chapter 15. Terminal COM class

The **Terminal** COM class represents a 3270 terminal connection to a CICS server. A CICS connection is established when the **Connect** method is called. Methods can then be used to converse with a 3270 terminal application (often a BMS application) in the CICS server.

---

### Interface Selection

For visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as Ccl0Terminal
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Object Creation

You can create an object in 2 ways

```
set var = CreateObject("Ccl.Terminal")  
set var = New Ccl0Terminal
```

The preferred method in Visual Basic is the use of **New**. For VBScript, you can only use the **CreateObject** method.

---

### Methods

#### AlterSecurity

**AlterSecurity(*newUserId* as String,*newPassword* as String)**

*newPassword*

The new password to be given to *newUserId*.

*newUserId*

The new userid.

Allows you to re-define the userid and password for a terminal resource that has not been constructed with one, ie Signon incapable created terminals. The method can be called before you install a terminal. It will change the terminal definition and the new userid and password will be used for the terminal when install is called.

## COM class: Terminal

### CCSId

**CCSId()** as long

Returns a long showing the selected code page.

### ChangePassword

**ChangePassword(*newPassword* as String) as Object**

or

**ChangePassword(*newPassword* as String) as CclOSecAttr**

*newPassword*

The new password to be given

Allows a client application to change the password held in the terminal object and the password recorded by an external security manager for the userid held in the terminal object. The external security manager is assumed to be located in the server defined by the terminal object. A CclOSecAttr Object is returned if no errors occurred.

### Connect

**Connect(*servName* as String,  
          *devType* as String,  
          *nworkName* as String)**

*servName*

The name of the server with which you require to communicate. If a NULL string is provided, the default server system, defined in the Client initialization file, is assumed. The name is expanded to 8 characters by padding with blanks, if necessary.

*devType*

The name of the model terminal definition which the server uses to generate a terminal resource definition. If a NULL string is provided the default model is used. The name is expanded to 16 characters by padding with blanks, if necessary.

*nworkName*

The name of the terminal resource to be installed or reserved. The name is expanded to 8 characters by padding with blanks, if necessary. If a NULL string is supplied, the CICS server will allocate a name.

Establishes a 3270 communication to the specified CICS server.

**Devtype****Devtype() as String**

Returns the terminal device type as a string.

**Diagnose****Diagnose() as String**

Returns a character string which holds a description of the error returned by the most recent server call.

**Disconnect****Disconnect()**

This method will disconnect the terminal and delete it. If a transaction was running at the time, the transaction will be purged and the terminal deleted.

**DisconnectWithPurge****DisconnectWithPurge()**

This method will disconnect the terminal and delete it. If a transaction was running at the time, the transaction will be purged and the terminal deleted.

**DiscReason****DiscReason() as CclEndTermReasons**

This method will return an enumeration showing the reason the terminal has been disconnected. Possible values are shown in “CclTerminal EndTermReasons” on page 92.

**ExCode****Deprecated method**

Do not use this method in new applications. The method has been deprecated and is only provided for backwards compatibility.

**ExCode() as Integer**

or

**ExCode() as CclEPIExceptionCodes**

Returns a value which indicates the most recent condition code returned by the server.

## COM class: Terminal

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

### ExCodeText

#### Deprecated method

Do not use this method in new applications. The method has been deprecated and is only provided for backwards compatibility.

#### ExCodeText() as String

Returns a text string describing the most recent condition code returned by the server.

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

### Install

#### Install(session as Object, timeout as Integer)

or

#### Install(session as CclOSession, timeout as Integer)

*session*

The session object to be used by this terminal object.

*InstallTimeout*

A value in the range 0 through 3600, specifying the maximum time in seconds that installation of the terminal resource is allowed to take. A value of 0 means that no limit is set.

This method installs a non-connected terminal resource. An `cclInvalidState` error will be raised if the terminal is already installed.

### MakeSecurityDefault

#### MakeSecurityDefault()

Informs the client that the current userid and password for this object is to become the default for ECI and EPI requests passed to the server as specified in the construction of the Terminal object.

### NetName

#### NetName() as String

Returns the network name of the terminal.



**Password****Password() as String**

Returns a text string containing the current password for the userid associated with the terminal. The string will be empty if there is no password.

**Poll****Poll() as Boolean**

Checks to see if a replies have been received from a deferred synchronous **Start** or **Send** request. Possible values are:

**True** No further replies outstanding

**False** Further replies outstanding

A CICS server transaction may send more than one reply in response to a **Terminal.Start** or **Terminal.Send** call. More than one **Terminal.Poll** call may therefore be needed to collect all the replies. The return code indicates whether you need to perform more poll requests.

**PollForReply****PollForReply() as Boolean**

Checks to see if replies have been received from a deferred synchronous **Start** or **Send** request. Possible values are

**true** Replies have been received

**false** No replies have been received

A CICS server transaction may send more than one reply in response to a **Terminal.Start** or **Terminal.Send** call. More than one **Terminal.PollForReply** call may therefore be needed to collect all replies. Use the **Terminal.State** method to find out if further replies are expected. If there are, the value returned will be **cclServer**.

**QueryATI****QueryATI() as Integer**

or

**QueryATI() as CclATISStates**

Returns a value which indicates whether the “Automatic Transaction Initiation” (ATI) is enabled or disabled. Possible values are:

**cclATIEnabled**

**cclATIDisabled**

## COM class: Terminal

### ReadTimeout

**ReadTimeout() as Integer**

Returns the read timeout setting for the terminal.

### ReceiveATI

**ReceiveATI (session as Object)**

or

**ReceiveATI (session as CclOSession)**

*session*

A pointer to the CclOSession object which is to be used for the CICS server interaction.

Waits for and receives 3270 datastream for a CICS ATI transaction. The CclOSession object supplied can only be synchronous.

### Screen

**Screen() as Object**

Returns the CclOScreen object that is handling the 3270 screen associated with this terminal.

### Send

**Send(session as Object)**

or

**Send(session as CclOSession)**

*session*

The CclOSession object which controls the session which is to be used. It is set to NULL if no CclOSession object is used.

Generates a 3270 data stream from the current contents of the **CclOScreen** object and transmits it to the CICS server.

### ServerName

**ServerName() as String**

Returns the name of the server system held by the CclOTerminal object and listed by the Client initialization file, or blanks if the default server is being used and no calls have yet been made.

**SetATI****SetATI(*stateVal* as Integer)**

or

**SetATI(*stateVal* as CclATISStates)***stateVal*

A value which indicates whether the ATI is to be enabled or disabled.

Possible values are:

cclATIEEnabled

cclATIDisabled

**SetTermDefns**

**SetTermDefns (*servName* as String,  
*devType* as String,  
*nworkName* as String  
*signonCapability* as CclSignonTypes  
*userid* as String  
*password* as String  
*ReadTimeout* as Integer  
*CCSid* as Long)**

*servName*

The name of the server with which you require to communicate. If a NULL string is provided, the default server system, defined in the Client initialization file, is assumed. The name is expanded to 8 characters by padding with blanks, if necessary.

*devType*

The name of the model terminal definition which the server uses to generate a terminal resource definition. If a NULL string is provided the default model is used. The name is expanded to 16 characters by padding with blanks, if necessary.

*nworkName*

The name of the terminal resource to be installed or reserved. The name is expanded to 8 characters by padding with blanks, if necessary. If a NULL string is supplied, the CICS server will allocate a name.

*signonCapability*

Set the signon capability to one of the following:

**cclSignonCapable****cclSignonIncapable**

## COM class: Terminal

### *ReadTimeout*

A value in the range 0 through 3600, specifying the maximum time in seconds between the time the classes go clientrepl state and the time that the application program invokes the reply method.

*userid* The name of the userid to associate with this terminal resource.

### *password*

The password to associate with the userid.

*CCSid* A long specifying the coded character set indentifier(CCSID) that identifies the coded graphic character set used by the client application for data passed between the terminal resource and CICS transactions. A zero implies a default will be used.

Creates a terminal resource but doesn't make the connection to the Server.

## SignonCapability

### **SignonCapability() as Integer**

or

### **SignonCapability() as CclSignonTypes**

Returns the type of terminal installed. Possible values are:

- cclSignonCapable
- cclSignonIncapable

## Start

**Start** (*session* as **Object**,  
          *tranCode* as **String**,  
          *startData* as **String**)

or

**Start** (*session* as **CclOSession**,  
          *tranCode* as **String**,  
          *startData* as **String**)

### *session*

The CclOSession object which controls the session which is to be used. It is set to NULL if no CclOSession object is used.

### *tranCode*

The name of the transaction which is to be started

### *startData*

Start transaction data. A NULL value indicates no data is required for the transaction being started.

Generates a 3270 data stream from the supplied data and transmits it to the CICS server, starting the named transaction.

## **State**

**State()** as Integer

or

**State()** as CclEPIStates

Returns a value which indicates the current state of the session. These values are the same as those returned by the **state** method in the **Session** COM class.

Constants are available in the type library. Use the Visual Basic Object Browser to view them.

## **TermId**

**TermId()** as String

Returns the terminal ID.

## **TransId**

**TransId()** as String

Returns the 4-character name of the current CICS transaction. Note that if a RETURN IMMEDIATE is run from the current transaction, TransId will not provide the name of the new transaction, it will still contain the name of the first transaction.

## **Userid**

**Userid()** as String

Returns a text string containing the current userid for the terminal. The string will be empty if there is no userid.

## **VerifyPassword**

**VerifyPassword()** as Object

or

**VerifyPassword()** as CclOSecAttr

Allows a client application to verify that the password held in the terminal object matches the password recorded by an external security manager for the userid held in the terminal object. The external security manager is assumed to be located in the server defined by the terminal object. A CclOSecAttr Object is returned if no errors occurred.



---

## Chapter 16. UOW COM class

Use this COM class when you make updates to recoverable resources in the server within a “unit of work” (UOW). Each update in a UOW is identified by a reference to its CcLOUOW object — see **Link** method in **Connect** COM class (“Link” on page 39).

---

### Interface Selection

For Visual Basic, the following types of interfaces are available as follows

```
Dim var as Object  
Dim var as CcLOUOW
```

The preferred method is the second example.

If you do not dim a variable or dim it with no type or you are using VBScript then the variable is assumed to be of type **Object**.

---

### Object Creation

You can create an object in 2 ways

```
set var = CreateObject("Ccl.UOW")  
set var = New CcLOUOW
```

The preferred method in Visual Basic is the use of **New**. For VBScript, you can only use the **CreateObject** method.

---

### Methods

#### BackOut

**BackOut**(*flow* as Object)

or

**BackOut**(*flow* as CcLOFlow)

*flow*

The CcLOFlow object which is used to control the client-server call  
Terminate this UOW and back out all changes made to recoverable resources in the server.

## Commit

**Commit(*flow* as Object)**

or

**Commit(*flow* as CclOFlow)**

*flow*

The CclOFlow object which is used to control the client-server call  
Terminate this UOW and commit all changes made to recoverable resources in the server.

## ForceReset

**ForceReset()**

Makes this UOW inactive and resets it. The UOW is neither committed or backed out.

## UowId

**UowId() as long**

Returns the identifier of the UOW. A zero return indicates that the UOW is either complete or has not yet started, in other words it is inactive.



---

## Part 3. Appendixes



---

## Appendix A. COM Global Constants

Constants are provided in the type libraries for the CICS client COM libraries. The libraries can be found in CCLIECI.DLL and CCLIEPI.DLL.

If you are using Visual Basic, you can look at the definitions in the type libraries by using Visual Basic Object viewer or another type library viewer.

If you are using VBScript, you cannot access the enumerations defined in the type library so you have to use the numeric values provided here.

The exception code constants are listed in “Appendix D. Error Code References” on page 97.



---

## Appendix B. CICS Client ECI Constants

---

### Synchronization Types

*Table 3. Synchronization types*

VB Enumeration	Value	Description
cclSync	0	Synchronous call type
cclDsync	1	Deferred synchronous call type

---

### Flow status types

*Table 4. Flow status types*

VB Enumeration	Value	Description
cclInactive	0	Flow is inactive
cclLink	1	Flow is currently making a link call
cclBackout	2	Flow is currently backing out a UOW
cclCommit	3	Flow is currently committing a UOW
cclStatus	4	Flow is requesting status
cclChanged	5	Flow is requesting a status change
cclCancel	6	Flow is requesting a status cancel

---

### Connection Status Codes

*Table 5. Connection status code*

VB Enumeration	Value	Description
cclUnknown	0	The CICS server status is unknown
cclAvailable	1	The CICS server status is available
cclUnavailable	2	The CICS server status is unavailable



---

## Appendix C. CICS Client EPI Specific Constants

---

### Synchronization Types

*Table 6. Synchronization types*

VB Enumeration	Value	Description
cclSync	0	Synchronous call type
cclDsync	1	Deferred synchronous call type

---

### CcIEPI States

*Table 7. CcIEPI States*

VB Enumeration	Value	Description
cclEPIActive	0	EPI initialized OK
cclDiscon	1	EPI Terminated
cclEPIError	2	EPI failed to initialize, handle exception for more information

---

### CclSession States

*Table 8. CclSession States*

VB Enumeration	Value	Description
cclSessionIdle	0	Idle, client needs to initiate transaction
cclSessionServer	1	Waiting for server
cclSessionClient	2	Waiting for client to respond
cclSessionDiscon	3	Disconnected
cclSessionError	4	Session Error, handle exception for more information

---

### CclTerminal States

*Table 9. CclTerminal States*

VB Enumeration	Value	Description
cclInit	0	Terminal defined but not installed
cclActive	1	Terminal connected (not used)
cclIdle	2	Idle, client needs to initiate transaction

Table 9. CclTerminal States (continued)

VB Enumeration	Value	Description
cclServer	3	Waiting for server
cclClient	4	Waiting for client to respond
cclDiscon	5	Disconnected
cclError	6	Terminal error, handle exception for more information

## CclTerminal ATI States

Table 10. CclTerminal ATI states

VB Enumeration	Value	Description
cclATIEnabled	0	ATIs are allowed
cclATIDisabled	1	ATIs are not allowed

## CclTerminal EndTermReasons

Table 11. CclTerminal ATI states

VB Enumeration	Value	Description
cclSignoff	0	Disconnect request or user has signed off the terminal
cclShutdown	1	The CICS server has been shut down
cclOutOfService	2	The terminal has been switched to out of use
cclUnknown	3	An unknown situation as occurred
cclFailed	4	The terminal failed to disconnect
cclNotDiscon	5	The terminal is not disconnected

## CclTerminal Signon Types

Table 12. CclTerminal Signon Types

VB Enumeration	Value	Description
cclSignonCapable	0	Terminal supports signon transaction
cclSignonIncapable	1	Terminal does not support signon transaction
cclSignonUnknown	2	Terminal signon capability is unknown



## CclScreen AID key codes

Table 13. CclScreen AID key codes

VB Enumeration	Value	Description
cclEnter	0	Enter key
cclClear	1	Clear key
cclPA1	2	Program Attention key 1
cclPA2	3	Program Attention key 2
cclPA3	4	Program Attention key 3
cclPF1	5	Program Function key 1
cclPF2	6	Program Function key 2
cclPF3	7	Program Function key 3
cclPF4	8	Program Function key 4
cclPF5	9	Program Function key 5
cclPF6	10	Program Function key 6
cclPF7	11	Program Function key 7
cclPF8	12	Program Function key 8
cclPF9	13	Program Function key 9
cclPF10	14	Program Function key 10
cclPF11	15	Program Function key 11
cclPF12	16	Program Function key 12
cclPF13	17	Program Function key 13
cclPF14	18	Program Function key 14
cclPF15	19	Program Function key 15
cclPF16	20	Program Function key 16
cclPF17	21	Program Function key 17
cclPF18	22	Program Function key 18
cclPF19	23	Program Function key 19
cclPF20	24	Program Function key 20
cclPF21	25	Program Function key 21
cclPF22	26	Program Function key 22
cclPF23	27	Program Function key 23
cclPF24	28	Program Function key 24

---

## CclField Protected State Attributes

Table 14. CclField Protected state attributes

VB Enumeration	Value	Description
cclProtect	0	Protected Field (cannot be modified)
cclUnprotect	1	Unprotected (input) field

---

## CclField Numeric Attributes

Table 15. CclField Numeric Attributes

VB Enumeration	Value	Description
cclAlphanumeric	0	Alphanumeric input field
cclNnumeric	1	Numeric input field

---

## CclField Intensity Attributes

Table 16. CclField Intensity attributes

VB Enumeration	Value	Description
cclNormal	0	Normal display
cclIntense	1	Intensified display
cclDark	2	Non-display field

---

## CclField Modified Attributes

Table 17. CclField Modified Attributes

VB Enumeration	Value	Description
cclUnmodified	0	Field has not been changed
cclModified	1	Field has been changed

---

## CclField Highlight Attributes

Table 18. CclField Highlight attributes

VB Enumeration	Value	Description
cclHltDefault	0	Default field text highlighting
cclHltNormal	1	Field text highlight as specified by 3270 base attribute
cclHltBlink	2	Blinking text
cclHltReverse	3	Reverse video text

Table 18. CclField Highlight attributes (continued)

VB Enumeration	Value	Description
cclHltUnderscore	4	Underscored text
cclHltIntense	5	High intensity text

## CclField Transparency Attributes

Table 19. CclField Transparency attributes

VB Enumeration	Value	Description
cclTrnDefault	0	Default (opaque) field background
cclTrnOr	1	Transparent field background (OR)
cclTrnXor	2	Transparent field background (XOR)
cclTrnOpaque	3	Opaque field background

## CclField Color Attributes

Table 20. CclField Color attributes

VB Enumeration	Value	Description
cclDefaultColor	0	
cclBlue	1	
cclRed	2	
cclPink	3	
cclGreen	4	
cclCyan	5	
cclYellow	6	
cclNeutral	7	
cclBlack	8	
cclDarkBlue	9	
cclOrange	10	
cclPurple	11	
cclPaleGreen	12	
cclPaleCyan	13	
cclGray	14	
cclWhite	15	



---

## Appendix D. Error Code References

Enumeration	Value	Description	ECI	EPI
cclNoError	0	No error occurred	Yes	Yes
cclBufferOverflow	1	Attempted to increase a CclBuf object which isn't Extensible	Yes	
cclMultipleInstance	2	Attempted to create more than one ECI object	Yes	
cclActiveFlow	3	Current Flow is still active, you cannot use this flow until it is inactive	Yes	
cclActiveUOW	4	Current UOW is still active, you need to backout or commit.	Yes	
cclSyncType	5	Incorrect synchronisation type for method call.	Yes	Yes
cclDataLength	9	CommArea > 32768 Bytes or inbound 3270 data stream too large for Terminal Buffer size.	Yes	Yes
cclNoCICS	10	The client is unavailable, or the server implementation is unavailable, or a logical unit of work was to be begun, but the CICS server specified is not available. No resources have been updated	Yes	Yes
cclCICSDied	11	A logical unit of work was to be begun or continued, but the CICS server was no longer available. If this is a link call with an active UOW, the changes are backed out. If This was a UOW Commit or the application cannot determine whether the changes have been committed or backed out, and must log this condition to aid future manual recovery	Yes	
cclNoReply	12	There was no outstanding reply	Yes	
cclTransaction	13	ECI Program Abended	Yes	
cclSystemError	14	Unknown internal error occurred	Yes	Yes

Enumeration	Value	Description	ECI	EPI
cclResource	15	The server implementation or the client did not have enough resources to complete the request e.g. insufficient SNA sessions.	Yes	Yes
cclMaxUOWs	16	A new logical unit of work was being created, but the application already has as many outstanding logical units of work as the configuration will support.	Yes	
cclUnknownServer	17	The requested server could not be located	Yes	Yes
cclSecurity	18	You did not supply a valid combination of user ID and password, though the server expects it.	Yes	Yes
cclMaxServers	19	You attempted to start requests to more servers than your configuration allows. You should consult the documentation for your client or server to see how to control the number of servers you can use.	Yes	Yes
cclMaxRequests	20	There were not enough communication resources to satisfy the request. You should consult the documentation for your client or server to see how to control communication resources	Yes	Yes
cclRolledBack	21	An attempt was made to commit a logical unit of work, but the server was unable to commit the changes, and backed them out instead	Yes	
cclParameter	22	Incorrect parameter supplied	Yes	Yes
cclInvalidState	23	The Object is not in the correct state to invoke the method, e.g. terminal object still in server state and an attempt to send data is made.	Yes	Yes
ccltransId	24	Null transid supplied or returned for a Pseudo Conversational transaction		Yes
cclInitEPI	25	No EPI object or EPI failed to initiliasse correctly		Yes

Enumeration	Value	Description	ECI	EPI
cclConnect	26	Unexpected error trying to add the terminal		Yes
cclDataStream	27	Unsupported Data Stream		Yes
cclInvalidMap	28	Map definition and Screen do not match		Yes
cclClass	29	Unknown internal Class error occurred.	Yes	Yes
cclStartTranFailure	30	Transaction failed to start		Yes
cclTimeout	31	Timeout occurred before response from Server	Yes	Yes
cclNoPassword	32	The object's password is null.	Yes	Yes
cclNoUserid	33	The object's userid is null	Yes	Yes
cclNullNewPassword	34	The provided password is null	Yes	Yes
cclPemNotSupported	35	The CICS Server doesn't support the Password Expiry Management facilities. The method cannot be used	Yes	Yes
cclPemNotActive	36	Password Expiry Management is not active	Yes	Yes
cclPasswordExpired	37	The password has expired. No information has been returned	Yes	Yes
cclPasswordInvalid	38	The password is invalid.	Yes	Yes
cclPasswordRejected	39	Change password failed because the password doesn't conform to standards defined	Yes	Yes
cclUseridInvalid	40	The userid is unknown	Yes	Yes
cclInvalidTermid	41	Invalid Terminal ID		Yes
cclInvalidModelId	42	Invalid Model/Type		Yes
cclnot3270	43	Not a 3270 device		Yes
cclinvalidCCSid	44	Invalid CCSid		Yes
cclServerBusy	45	CICS server is busy		Yes
cclSignonNotPoss	46	The server does not allow the terminal to be installed as signon capable.		Yes





---

## Notices

This information was developed for products and services offered in the U.S.A. IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

IBM Director of Licensing  
IBM Corporation  
North Castle Drive  
Armonk, NY 10504-1785  
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation  
Licensing  
2-31 Roppongi 3-chome, Minato-ku  
Tokyo 106, Japan

**The following paragraph does not apply in the United Kingdom or any other country where such provisions are inconsistent with local law:**  
INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact IBM United Kingdom Laboratories, MP151, Hursley Park, Winchester, Hampshire, England, SO21 2JN. Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

#### COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. You may copy, modify, and distribute these sample programs in any form without payment to IBM for the purposes of developing, using, marketing, or distributing application programs conforming to IBM's application programming interfaces.

---

## Trademarks and service marks

The following terms, used in this publication, are trademarks or service marks of IBM Corporation in the United States or other countries:

CICS

IBM

Microsoft, Windows, Windows NT and the Windows logo are trademarks of Microsoft Corporation.

UNIX is a registered trademark in the United States and other countries licensed exclusively through X/Open Company Limited.



---

## Bibliography

### **The CICS Transaction Gateway and CICS Universal Clients library**

This chapter lists all the CICS Transaction Gateway, CICS Universal Clients, and related books, and discusses the various forms in which they are available.

The headings in this chapter are:

- “CICS Transaction Gateway books”
- “CICS Universal Clients books”
- “CICS Family publications” on page 106
- “Book filenames” on page 106
- “Sample configuration documents” on page 106
- “Other publications” on page 107
- “Viewing the online documentation” on page 107

### **CICS Transaction Gateway books**

- *CICS Transaction Gateway for OS/2 Administration*, SC34-5590

This book describes the administration of the CICS Transaction Gateway for OS/2.

- *CICS Transaction Gateway for Windows Administration*, SC34-5589

This book describes the administration of CICS Transaction Gateway for Windows 98 and CICS Transaction Gateway for Windows NT.

- *CICS Transaction Gateway for AIX Administration*, SC34-5591

This book describes the administration of the CICS Transaction Gateway for AIX.

- *CICS Transaction Gateway for Solaris Administration*, SC34-5592

This book describes the administration of the CICS Transaction Gateway for Solaris.

- *CICS Transaction Gateway for OS/390 Administration*, SC34-5528

This book describes the administration of the CICS Transaction Gateway for OS/390.

- *CICS Transaction Gateway Messages*

This online book lists and explains the error messages that can be generated by CICS Transaction Gateway.

You cannot order this book.

- *CICS Transaction Gateway Programming*, SC34-5594

This book provides an introduction to Java programming with the CICS Transaction Gateway.

There are also additional HTML pages that contain programming reference information.

### **CICS Universal Clients books**

- *CICS Universal Client for OS/2 Administration*, SC34-5450

This book describes the administration of the CICS Universal Client for OS/2.

- *CICS Universal Client for Windows Administration*, SC34-5449

This book describes the administration of the CICS Universal Client for Windows 98 and CICS Universal Client for Windows NT.

- *CICS Universal Client for AIX Administration*, SC34-5348

This book describes the administration of the CICS Universal Client for AIX.

- *CICS Universal Client for Solaris Administration*, SC34-5451

This book describes the administration of the CICS Universal Client for Solaris.

## The CICS Transaction Gateway and CICS Universal Clients library

- *IBM CICS Universal Clients Messages*

This online book lists and explains the error and trace messages that can be generated by CICS Universal Clients.

You cannot order this book.

- *IBM CICS Universal Clients C++ Programming*, SC33-1923

This book describes how to write object oriented programs for the ECI and EPI in the C++ language.

- *IBM CICS Universal Clients COM Automation Programming*, SC33-1924

This book describes how to write object oriented programs for the ECI and EPI according to the Component Object Model (COM) standard.

### CICS Family publications

- *CICS Family: Client/Server Programming*, SC33-1435

This book describes the programming interfaces associated with CICS client/server Programming—the External Call Interface (ECI), the External Presentation Interface (EPI), and the External Security Interface (ESI). It is intended for application designers and programmers who wish to develop client applications to communicate with CICS server systems.

### Book filenames

Table 21 show the softcopy filenames of the CICS Transaction Gateway and CICS Universal Client books.

Table 21. CICS Transaction Gateway and CICS Universal Clients books and file names

Book title	File name
IBM CICS Universal Clients Messages	CCLHAB
CICS Universal Client for AIX Administration	CCLHAD
CICS Universal Client for OS/2 Administration	CCLHAE
CICS Universal Client for Windows Administration	CCLHAF
CICS Universal Client for Solaris Administration	CCLHAG
CICS Transaction Gateway for OS/390 Administration	CCLHAI
CICS Transaction Gateway Messages	CCLHAJ
CICS Transaction Gateway Programming	CCLHAK
CICS Transaction Gateway for Windows Administration	CCLHAL
CICS Transaction Gateway for OS/2 Administration	CCLHAM
CICS Transaction Gateway for AIX Administration	CCLHAN
CICS Transaction Gateway for Solaris Administration	CCLHAO
IBM CICS Universal Clients C++ Programming	CCLHAP
IBM CICS Universal Clients COM Automation Programming	CCLHAQ
CICS Family: Client/Server Programming	DFHZAD
<b>Note:</b> The File names in this table do not include the 2-digit suffix.	

### Sample configuration documents

A number of sample configuration documents are available in the Portable Document Format (PDF) format.

These documents provide step-by-step guidance to help you, for example, in configuring your CICS Universal Clients for communication with CICS servers, using various protocols. They provide detailed instructions that extend the information in the CICS Transaction Gateway and CICS Universal Client libraries.

As more sample configuration documents become available, you can download them from our Web site; go to:

<http://www.ibm.com/software/ts/cics/>

and follow the **Library** link.

### Other publications

The following International Technical Support Organization (ITSO) Redbook publication contains many examples of client/server configurations:

- *Revealed! CICS Transaction Gateway with more CICS Clients Unmasked, SG24-5277*

This book supersedes the following book:

- *CICS Clients Unmasked, GG24-2534*

You can obtain ITSO Redbooks from a number of sources. For the latest information, see:

<http://www.ibm.com/redbooks/>

You can find information on CICS products at:

<http://www.ibm.com/software/ts/cics/>

### Viewing the online documentation

You can access all of the documentation provided with CICS Transaction Gateway and CICS Universal Client in our online library. You need Adobe Acrobat Reader and a suitable Web browser to use the online library (and you may need to configure these).

To get to the online library:

- On Windows and OS/2, select the **Documentation** icon
- On AIX and Solaris, run the **ctgdoc** script and the library home page is displayed.

The online library allows you to link to:

- CICS Transaction Gateway and CICS Universal Clients books in PDF format.
- Programming reference documentation in HyperText Markup Language (HTML) files (provided for CICS Transaction Gateway only).
- README files.
- Sample configuration documents in PDF format.
- Translated books in PDF format. (You may find that not all books are translated for your language.)
- The CICS Web site.

Guidance information on using Acrobat Reader is also provided.

Updated versions of the books may be provided from time to time, check our Web site at:

<http://www.ibm.com/software/ts/cics/>

and follow the **Library** link.

### Viewing PDF books

The PDF information provides powerful functions for:

- Navigating through the information. There are hypertext links within PDF documents, and to other PDF documents and Web pages.
- Searching for specific information.
- Printing all or part of PDF documents on a PostScript printer.

## Viewing the online documentation

- | You can find out more about Acrobat
- | Reader at the Adobe Web site:
- | <http://www.adobe.com/acrobat/>



---

## Glossary

**AID.** Attention Identifier

**ATI.** Automatic Transaction Initiation

**BMS.** Basic Mapping Support

**COMMAREA.** CICS Communications Area

**COM.** Component Object Model

**ECI.** External Call Interface

**EPI.** External Presentation Interface

**ESI.** External Security Interface

**ISC.** Inter-Systems Communication

**OLE.** Object Linking and Embedding

**UOW.** Unit of Work



---

# Index

## Numerics

3270 datastreams 20

## A

AbendCode  
    in Methods  
    in Flow COM Class 57  
AlterSecurity  
    in Methods  
    in Connect COM Class 37  
    in Terminal COM Class 73  
AppendString  
    in Methods  
    in Buffer COM Class 34  
AppendText  
    in Methods  
    in Field COM Class 51  
array (parameter)  
    in SetData 35  
ATI 25  
attachTran (parameter)  
    in TranDetails 41  
Attribute (parameter)  
    in SetBaseAttribute 54  
    in SetExtAttribute 55  
Automatic Transaction Initiation 25  
automation compatible 5

## B

BackgroundColor  
    in Methods  
    in Field COM Class 51  
Backout  
    in ECI Link Calls within a Unit  
    Of Work 18  
    in Poll 58  
BackOut  
    in Methods  
    in UOW COM Class 83  
BaseAttribute  
    in Methods  
    in Field COM Class 52  
books 105  
    CICS Transaction Gateway and  
    CICS Universal Clients  
    library 105  
    online 107  
    PDF 107  
    printed 108

Buffer  
    in AppendString 34  
    in Buffer COM Class 33  
    in Link 40  
    in Poll 58  
    in String 35  
Buffer COM class  
    Methods  
        AppendString 34  
        Data 34  
        ExtractString 34  
        InsertString 34  
        Length 34  
        Overlay 35  
        SetData 35  
        SetLength 35  
        SetString 35  
        String 35  
business object 15

## C

CallType  
    in Methods  
    in Flow COM Class 57  
CallTypeText  
    in Methods  
    in Flow COM Class 58  
Cancel  
    in Methods  
    in Connect COM Class 38  
    in Poll 58  
Ccal Screen.fieldbyPosition method  
    in Field COM class 51  
Ccl Field  
    in Running a CICS 3270  
    session 21  
Ccl.Field  
    in FieldByName 62  
Ccl Map  
    in Using BMS Map data with EPI  
    COM classes 25  
Ccl.Screen  
    in Send 78  
cclActive  
    in State 50, 72  
cclAlphanumeric  
    in InputType 53  
cclATIDisabled  
    in QueryATI 77  
    in SetATI 79  
cclATIEnabled  
    in QueryATI 77  
    in SetATI 79  
cclAvailable  
    in ServerStatus 40  
cclClient  
    in State 72  
cclDark  
    in Intensity 54  
cclDiscon  
    in State 50, 72  
cclDSync  
    in ECI Call Synchronization  
    Types 16  
    in EPI call synchronization  
    types 22  
    in SetSyncType 59, 72  
cclError  
    in State 50, 72  
cclIntense  
    in Intensity 54  
cclModified  
    in DataTag 52  
cclNoError  
    in ExCode 48  
cclNormal  
    in Intensity 54  
cclNumeric  
    in InputType 53  
CCLOECLEXE  
    in Using the COM classes 9  
CCLOEPIEXE  
    in Using the COM classes 9  
CclOSecTime  
    in Ccl SecAttr interface 68  
cclProtect  
    in InputProt 53  
cclServer  
    in State 72  
cclSync  
    in ECI Call Synchronization  
    Types 16  
    in EPI call synchronization  
    types 22  
    in SetSyncType 59, 71, 72  
cclSystemError  
    in ExCode 48  
cclUnavailable  
    in ServerStatus 40

- cclUnknown
  - in ServerStatus 40
- cclUnknownServer
  - in ExCode 48
- cclUnmodified
  - in DataTag 52
  - in ResetDataTag 54
- cclUnprotect
  - in InputProt 53
- CCSID
  - in Methods
    - in Terminal COM class 74
- CCSID (parameter)
  - in SetTermDefns 79
- changed
  - in ServerStatus 40
  - in ServerStatusText 41
- Changed
  - in Cancel 38
  - in Changed 38
  - in Details 39
  - in Methods
    - in Connect COM Class 38
  - in Poll 58
- ChangePassword
  - in Methods
    - in Connect COM Class 38
    - in Terminal COM Class 74
- CICS Server Information
  - in Connecting to CICS 3270 applications using the EPI
  - in Using the COM classes 23
- CICS Server Information and Connection Status
  - in Making an ECI link call to CICS
  - in Using the COM classes 17
- Client initialization file
  - in Connect 74
  - in Details 39
  - in ServerCount 45, 49
  - in ServerDesc 49
  - in ServerName 40, 49, 78
- colPos (parameter)
  - in FieldByPosition 64
  - in SetCursor 65
- Column
  - in Methods
    - in Field COM Class 52
- commArea (parameter)
  - in AbendCode 57
  - in Link 39, 40
  - in Poll 58
- Commit
  - in ECI Link Calls within a Unit Of Work 18
  - in Methods
    - in UOW COM Class 84
  - in Poll 58
- Component Object Model
  - in Establishing the working environment 5
- Connect
  - in CICS Server Information and Connection Status 17
  - in Connect COM Class 37
  - in Methods
    - in Terminal COM Class 74
  - in ServerName 40
  - in Terminal COM class 73
  - in UOW COM Class 83
  - in UserId 42
- Connect COM class
  - Methods
    - AlterSecurity 37
    - Cancel 38
    - Changed 38
    - ChangePassword 38
    - Details 39
    - Link 39
    - MakeSecurityDefault 40
    - Password 40
    - ServerName 40
    - ServerStatus 40
    - ServerStatusText 41
    - Status 41
    - TranDetails 41
    - UnpaddedPassword 41
    - UnpaddedServerName 42
    - UnpaddedUserId 42
    - UserId 42
    - VerifyPassword 42
- Connect.Link
  - in ECI Call Synchronization Types 16
  - in ECI Link Calls within a Unit Of Work 18
- Connecting to CICS 3270 applications using the EPI
  - CICS Server Information 23
  - EPI call synchronization types 22
  - in Using the COM classes 20
  - Running a CICS 3270 session 21
  - Using BMS Map data with EPI COM classes 23
- CreateObject 33, 37, 43, 47, 57, 61, 71, 73, 83
  - in Connecting to CICS 3270 applications using the EPI 20
  - in Making an ECI link call to CICS using VBScript 16
- CursorCol
  - in Methods
    - in Screen COM Class 63
- CursorRow
  - in Methods
    - in Screen COM Class 63
- D**
- Data
  - in Methods
    - in Buffer COM Class 34
- DataTag
  - in Methods
    - in Field COM Class 52
- Day
  - in SecTime COM Class 69
- DBCS 20
- Deferred synchronous
  - in ATI support 26
- Delphi 5
- Depth
  - in Methods
    - in Screen COM Class 63
- Details
  - in Connect COM class 37
  - in Making an ECI link call to CICS 15
  - in Methods
    - in Connect COM Class 39
- Devtype
  - in Methods
    - in Terminal COM Class 75
- devType (parameter)
  - in Connect 74
  - in SetTermDefns 79
- Diagnose
  - in EPI COM Class 47
  - in Methods
    - in EPI COM Class 47
    - in Flow COM Class 58
    - in Session COM Class 71
    - in Terminal COM Class 75
  - in State 50, 72
- Disconnect
  - in Disconnect 75
  - in Methods
    - in Terminal COM Class 75

- DisconnectWithPurge
  - in Methods
  - in Terminal COM Class 75
- DiscReason
  - in DiscReason 75
  - in Methods
  - in Terminal COM Class 75
- display (parameter)
  - in ErrorWindow 44, 48
- documentation 105
  - HTML 107
  - PDF 107

## E

- ECI
  - in CICS Server Information and Connection Status 17
- ECI Call Synchronization Types
  - in Making an ECI link call to CICS
  - in Using the COM interfaces 16
- ECI COM class
  - Methods
    - ErrorFormat 43
    - ErrorOffset 43
    - ErrorWindow 44
    - ExCode 44
    - ExCodeText 44
    - ServerCount 45
    - ServerDesc 45
    - ServerName 45
    - SetErrorFormat 45
- ECI Link Calls within a Unit Of Work
  - in Making an ECI link call to CICS
  - in Using the COM interfaces 18
- Environments supported
  - in Establishing the working environment 5
- EPI
  - in CICS Server Information 23
  - in EPI COM Class 47
- EPI call synchronization types
  - in Connecting to CICS 3270 applications using the EPI
  - in Using the COM classes 22
- EPI COM class
  - Methods
    - Diagnose 47
    - ErrorFormat 47
    - ErrorOffset 48
    - ErrorWindow 48

## EPI COM class *(continued)*

- Methods *(continued)*
  - ExCode 48
  - ExCodeText 49
  - ServerCount 49
  - ServerDesc 49
  - ServerName 49
  - SetErrorFormat 49
  - State 50
  - Terminate 50
- Err objects 12
- ErrorFormat
  - in Methods
  - in ECI COM Class 43
  - in EPI COM Class 47
- ErrorOffset
  - in Methods
  - in ECI COM Class 43
  - in EPI COM Class 48
- ErrorWindow 12
  - in Methods
  - in ECI COM Class 44
  - in EPI COM Class 48
- exception handling 12
- ExCode 12, 44, 48
  - in EPI COM Class 47
  - in ErrorWindow 44, 48
  - in Methods
    - in ECI COM Class 44
    - in EPI COM Class 48
    - in Map COM Class 61
    - in Terminal COM Class 75
  - in State 50, 72
- ExCodeText 12
  - in ExCode 44
  - in Methods
    - in ECI COM Class 44
    - in EPI COM Class 49
    - in Terminal COM Class 76
- ExpiryTime
  - in SecAttr COM class 67
- ExtractString
  - in Methods
  - in Buffer COM Class 34

## F

- false
  - in ErrorWindow 44, 48
- False
  - in Poll 58, 77
- FALSE
  - in Validate 62
- Field
  - in Field COM Class 51
  - in Screen COM Class 63

## Field COM class

- Methods
  - AppendText 51
  - BackgroundColor 51
  - BaseAttribute 52
  - Column 52
  - DataTag 52
  - ForegroundColor 52
  - Highlight 53
  - InputProt 53
  - InputType 53
  - Intensity 54
  - Length 54
  - Position 54
  - ResetDataTag 54
  - Row 54
  - SetBaseAttribute 54
  - SetExtAttribute 55
  - SetText 55
  - Text 55
  - TextLength 55
  - Transparency 55
- FieldByIndex
  - in Methods
  - in Screen COM Class 64
  - in Using BMS Map data with EPI COM classes 25
- FieldByName
  - in Methods
  - in Map COM Class 62
  - in Validate 62
- FieldByPosition
  - in Methods
  - in Screen COM Class 64
  - in Using BMS Map data with EPI COM classes 25
- FieldCount
  - in Methods
  - in Screen COM Class 64
- Flow
  - in Cancel 38
  - in Changed 38
  - in CICS Server Information and Connection Status 17
  - in Commit 84
  - in Connect COM Class 37
  - in Flow COM class 57
  - in Flow COM Class 57
  - in Link 39
  - in SetSyncType 59
  - in Status 41
- flow (parameter)
  - in BackOut 83
  - in Cancel 38
  - in Changed 38

flow (parameter) *(continued)*

- in Commit 84
- in Link 39
- in Status 41

Flow COM class

Methods

- AbendCode 57
- CallType 57
- CallTypeText 58
- Diagnose 58
- Flowid 58
- ForceReset 58
- Poll 58
- SetSyncType 59
- SetTimeout 59
- SyncType 59
- Timeout 59
- Wait 59

Flowid

- in Methods
- in Flow COM Class 58

ForceReset

- in Methods
- in Flow COM Class 58
- in UOW COM Class 84

ForegroundColor

- in Methods
- in Field COM Class 52

Form\_Load

- in Connecting to CICS 3270 applications using the EPI 20
- in Making an ECI link call to CICS 15

format (parameter)

- in SetErrorFormat 45, 49

## G

GetDate

- in SecTime COM Class 69

## H

handling, exception 12

hardcopy books 108

Highlight

- in Methods
- in Field COM Class 53

Hours

- in SecTime COM Class 69

HTML (HyperText Markup Language) 107

HTML documentation, viewing 107

Hunderdths

- in SecTime COM Class 69

HyperText Markup Language (HTML) 107

## I

in Buffer COM Class 33

in Connect COM Class 37

in ECI COM Class 43

in EPI COM Class 47

in field COM Class 57, 61, 71, 73, 83

index (parameter)

- in ExCode 48
- in FieldByIndex 64
- in ServerDesc 45, 49
- in ServerName 45, 49

InputProt

- in Methods
- in Field COM Class 53

InputType

- in Methods
- in Field COM Class 53

InsertString

- in Methods
- in Buffer COM Class 34

Install

- in Methods
- in Terminal COM Class 76

Intensity

- in Methods
- in Field COM Class 54

## K

key (parameter)

- in SetAID 65

## L

LastVerifiedTime

- in SecAttr COM class 68

Length

- in Methods
- in Buffer COM Class 34
- in Field COM Class 54

length (parameter)

- in ExtractString 34
- in SetLength 35

link

- in SetSyncType 59

Link

- in Details 39
- in Making an ECI link call to CICS 15
- in Methods
- in Connect COM Class 39
- in Poll 58
- in UOW COM class 83

## M

MakeSecurityDefault

- in Methods
- in Connect COM Class 40
- in Terminal COM Class 76

Making an ECI link call to CICS  
CICS Server Information and

- Connection Status 17
- ECI Call Synchronization Types 16

ECI Link Calls within a Unit Of  
Work 18

- in Using the COM classes 14

Map

- in Map COM Class 61
- in Using BMS Map data with EPI COM classes 23, 24

Map COM class

Methods

- ExCode 61
- FieldByName 62
- Validate 62

Map.FieldByName

- in Using BMS Map data with EPI COM classes 25

Map.Validate

- in Using BMS Map data with EPI COM classes 25

MapName

- in Methods
- in Screen COM Class 64

mapname (parameter)

- in Validate 62

MapSetName

- in Methods
- in Screen COM Class 64

Methods

- AbendCode 57
- AlterSecurity 37, 73
- AppendString 34
- AppendText 51
- BackgroundColor 51
- BackOut 83
- BaseAttribute 52
- CallType 57
- CallTypeText 58
- Cancel 38
- CCSID 74
- Changed 38
- ChangePassword 38, 74
- Column 52
- Commit 84
- Connect 74
- CursorCol 63
- CursorRow 63

## Methods (continued)

- Data 34
- DataRow 52
- Depth 63
- Details 39
- Devtype 75
- Diagnose 47, 58, 71, 75
- Disconnect 75
- DisconnectWithPurge 75
- DiscReason 75
- ErrorFormat 43, 47
- ErrorOffset 43, 48
- ErrorWindow 44, 48
- ExCode 44, 48, 61, 75
- ExCodeText 44, 49, 76
- ExtractString 34
- FieldByIndex 64
- FieldByName 62
- FieldByPosition 64
- FieldCount 64
- Flowid 58
- ForceReset 58, 84
- ForegroundColor 52
- Highlight 53
  - in Buffer COM Class 34
  - in Connect COM Class 37
  - in ECI COM Class 43
  - in EPI COM Class 47
  - in Field COM Class 51
  - in Flow COM Class 57
  - in Map COM Class 61
  - in Screen COM Class 63
  - in Session COM Class 71
  - in Terminal COM Class 73
  - in UOW COM Class 83
- InputProt 53
- InputType 53
- InsertString 34
- Install 76
- Intensity 54
- Length 34, 54
- Link 39
- MakeSecurityDefault 40, 76
- MapName 64
- MapSetName 64
- NetName 76
- Overlay 35
- Password 40, 77
- Poll 58, 77
- PollForReply 77
- Position 54
- QueryATI 77
- ReadTimeout 78
- ReceiveATI 78
- ResetDataRow 54

## Methods (continued)

- Row 54
- Screen 78
- Send 78
- ServerCount 45, 49
- ServerDesc 45, 49
- ServerName 40, 45, 49, 78
- ServerStatus 40
- ServerStatusText 41
- SetAID 65
- SetATI 79
- SetBaseAttribute 54
- SetCursor 65
- SetData 35
- SetErrorFormat 45, 49
- SetExtAttribute 55
- SetLength 35
- SetString 35
- SetSyncType 59, 71
- SetTermDefns 79
- SetText 55
- SetTimeout 59
- SignonCapability 80
- Start 80
- State 50, 72, 81
- Status 41
- String 35
- SyncType 59
- TermId 81
- Terminate 50
- Text 55
- TextLength 55
- Timeout 59
- TranDetails 41
- TransId 72, 81
- Transparency 55
- UnpaddedPassword 41
- UnpaddedServerName 42
- UnpaddedUserId 42
- UowId 84
- UserId 81
- UserId 42
- Validate 62
- VerifyPassword 42, 81
- Wait 59
- Width 65

Minutes

- in SecTime COM Class 70

Month

- in SecTime COM Class 70

## N

name (parameter)

- in FieldByName 62

## NetName

- in Methods
- in Terminal COM Class 76

## New

- in Buffer COM Class 33
- in Connect COM Class 37
- in ECI COM Class 43
- in EPI COM Class 47
- in Field COM Class 57, 61, 71, 73, 83
- in Making an ECI link call to CICS 15
- newPassword (parameter)
  - in AlterSecurity 37, 73
  - in ChangePassword 38
- newUserId (parameter)
  - in AlterSecurity 37, 73
- Nothing
  - in Poll 58
- nworkName (parameter)
  - in Connect 74
  - in SetTermDefns 79

## O

offset (parameter)

- in ExtractString 34
- in InsertString 34
- in Overlay 35

On Error Goto 12

On Error Resume 12

online books, PDF 107

online documentation, HTML 107

OO support in CICS Clients

- in Introduction to OO programming 3

Overlay

- in Methods
- in Buffer COM Class 35

## P

### Password

- in Methods
- in Connect COM Class 40
- in Terminal COM Class 77

### password (parameter)

- in ChangePassword 74
- in Details 39
- in SetTermDefns 79

### PDF (Portable Document Format) 107

### PDF books, viewing 107

### Poll

- in ECI Call Synchronization Types 16, 17

- Poll (*continued*)
  - in EPI call synchronization types 22
  - in Methods
    - in Flow COM Class 58
    - in Terminal COM Class 77
  - in SetSyncType 59, 72
- poll method
  - in ATI support 26
- PollForReply
  - in Methods
    - in Terminal COM Class 77
- pollForReply method
  - in ATI support 26
- Portable Document Format (PDF) 107
- Position
  - in Methods
    - in Field COM Class 54
- PostScript books 108
- Programming Overview
  - in Using the COM interfaces 11
- programName (parameter)
  - in Link 39
- Public Methods
  - in SecAttr COM class 67
  - in SecTime COM Class 69
- publications, CICS Transaction Gateway and CICS Universal Clients library 105

## Q

- queryATI 26
- QueryATI
  - in Methods
    - in Terminal COM Class 77

## R

- ReadTimeout
  - in Methods
    - in Terminal COM Class 78
- ReadTimeout (parameter)
  - in SetTermDefns 79
- ReceiveATI
  - in Methods
    - in Terminal COM Class 78
- receiveATI method
  - in ATI support 26
- Registration 6
- ResetDataTag
  - in Methods
    - in Field COM Class 54
    - in ResetDataTag 54
- Row
  - in Methods
    - in Field COM Class 54

- rowPos (parameter)
  - in FieldByPosition 64
  - in SetCursor 65
- Running a CICS 3270 session
  - in Connecting to CICS 3270 applications using the EPI
    - in Using the COM classes 21
- runTran (parameter)
  - in TranDetails 41

## S

- Screen
  - in Field COM Class 51
  - in Methods
    - in Terminal COM Class 78
  - in Running a CICS 3270 session 21
  - in Screen 78
  - in Screen COM class 63
  - in Using BMS Map data with EPI COM classes 25
  - in Validate 62
- Screen COM class
  - Methods
    - CursorCol 63
    - CursorRow 63
    - Depth 63
    - FieldByIndex 64
    - FieldByPosition 64
    - FieldCount 64
    - MapName 64
    - MapSetName 64
    - SetAID 65
    - SetCursor 65
    - Width 65
- Screen.fieldbyIndex method
  - in Ccl Field COM class 51
- screenRef (parameter)
  - in Validate 62
- Seconds
  - in SecTime COM Class 70
- Send
  - in Methods
    - in Terminal COM Class 78
  - in Poll 77
  - in SetSyncType 72
- send method
  - in ATI support 26
- ServerCount
  - in Methods
    - in ECI COM Class 45
    - in EPI COM Class 49
- ServerDesc
  - in ExCode 48

- ServerDesc (*continued*)
  - in Methods
    - in ECI COM Class 45
    - in EPI COM Class 49
- ServerName
  - in Details 39
  - in ExCode 48
  - in Methods
    - in Connect COM Class 40
    - in ECI COM Class 45
    - in EPI COM Class 49
    - in Terminal COM Class 78
- serverName (parameter)
  - in Details 39
- Servers 5
- ServerStatus
  - in Methods
    - in Connect COM Class 40
- ServerStatusText
  - in Methods
    - in Connect COM Class 41
- servName (parameter)
  - in Connect 74
  - in SetTermDefns 79
- Session
  - in Running a CICS 3270 session 21
  - in Send 78
  - in Session COM Class 71
  - in SetSyncType 71, 72
  - in Start 80
  - in State 81
- session (parameter)
  - in ReceiveATI 78
  - in Send 78
  - in Start 80
- Session COM class
  - Methods
    - Diagnose 71
    - SetSyncType 71
    - State 72
    - TransId 72
- Session.SetSyncType
  - in EPI call synchronization types 22
- SetAID
  - in Methods
    - in Screen COM Class 65
- setATI 26
- SetATI
  - in Methods
    - in Terminal COM Class 79
- SetBaseAttribute
  - in Methods
    - in Field COM Class 54



- SetCursor
  - in Methods
  - in Screen COM Class 65
- SetData
  - in Methods
  - in Buffer COM Class 35
- SetErrorFormat 12
  - in Methods
  - in ECI COM Class 45
  - in EPI COM Class 49
- SetExtAttribute
  - in Methods
  - in Field COM Class 55
- SetLength
  - in Buffer COM Class 33
  - in Methods
  - in Buffer COM Class 35
- SetString
  - in Methods
  - in Buffer COM Class 35
- SetSyncType
  - in ECI Call Synchronization Types 16
  - in Methods
  - in Flow COM Class 59
  - in Session COM Class 71
- SetTermDefns
  - in Methods
  - in Terminal COM Class 79
- SetText
  - in Methods
  - in Field COM Class 55
- SetTimeout
  - in Methods
  - in Flow COM Class 59
- SignonCapability
  - in Methods
  - in Terminal COM Class 80
- signonCapability (parameter)
  - in SetTermDefns 79
- softcopy books, PDF 107
- Start
  - in Methods
  - in Terminal COM Class 80
  - in Poll 77
  - in SetSyncType 72
- start method
  - in ATI support 26
- startData (parameter)
  - in Start 80
- State
  - in EPI COM Class 47
  - in Methods
  - in EPI COM Class 50
  - in Session COM Class 72

- State (*continued*)
  - in Methods (*continued*)
  - in Terminal COM Class 81
  - in State 81
- stateVal (parameter)
  - in SetATI 79
- status
  - in ServerStatus 40
  - in ServerStatusText 41
  - in SetSyncType 59
- Status
  - in Details 39
  - in Methods
  - in Connect COM Class 41
  - in Poll 58
- String
  - in Methods
  - in Buffer COM Class 35
- string (parameter)
  - in AppendString 34
  - in InsertString 34
  - in Overlay 35
  - in SetString 35
- Synchronous
  - in ATI support 26
- SyncType
  - in Methods
  - in Flow COM Class 59
- syncType (parameter)
  - in SetSyncType 59, 71
  - in SyncType 59

## T

- TermId
  - in Methods
  - in Terminal COM Class 81
- Terminal
  - in EPI COM Class 47
  - in Running a CICS 3270 session 21
  - in Screen COM Class 63
  - in ServerName 78
- Terminal class 26
- Terminal COM class
  - Methods
    - AlterSecurity 73
    - CCSID 74
    - ChangePassword 74
    - Connect 74
    - Devtype 75
    - Diagnose 75
    - Disconnect 75
    - DisconnectWithPurge 75
    - DiscReason 75
    - ExCode 75

- Terminal COM class (*continued*)
  - Methods (*continued*)
    - ExCodeText 76
    - Install 76
    - MakeSecurityDefault 76
    - NetName 76
    - Password 77
    - Poll 77
    - PollForReply 77
    - QueryATI 77
    - ReadTimeout 78
    - ReceiveATI 78
    - Screen 78
    - Send 78
    - ServerName 78
    - SetATI 79
    - SetTermDefns 79
    - SignonCapability 80
    - Start 80
    - State 81
    - TermId 81
    - TransId 81
    - Userid 81
    - VerifyPassword 81
- Terminal.Connect
  - in Screen COM Class 63
- Terminal.Poll
  - in EPI call synchronization types 22
- Terminal.Screen
  - in Screen COM Class 63
- Terminal.Send
  - in EPI call synchronization types 22
- Terminal.Start
  - in EPI call synchronization types 22
  - in Running a CICS 3270 session 21
- Terminate
  - in Methods
  - in EPI COM Class 50
  - in Terminate 50
- Text
  - in Methods
  - in Field COM Class 55
- TextLength
  - in Methods
  - in Field COM Class 55
- textString (parameter)
  - in AppendText 51
  - in SetText 55
- Timeout
  - in Methods
  - in Flow COM Class 59

- timeout (parameter)
  - in Install 76
- Trademarks and service marks
  - in Notices 103
- tranCode (parameter)
  - in Start 80
- TranDetails
  - in Connect COM Class 37
  - in Methods
    - in Connect COM Class 41
- Transaction Initiation, Automatic 25
- TransId
  - in Methods
    - in Session COM Class 72
    - in Terminal COM Class 81
- Transparency
  - in Methods
    - in Field COM Class 55
- true
  - in ErrorWindow 44, 48
- True
  - in Poll 58, 77
- TRUE
  - in Validate 62

## U

- unitOfWork (parameter)
  - in Link 39, 40
- UnpaddedPassword
  - in Methods
    - in Connect COM Class 41
- UnpaddedServerName
  - in Methods
    - in Connect COM Class 42
- UnpaddedUserId
  - in Methods
    - in Connect COM Class 42
- UOW
  - in ECI Link Calls within a Unit Of Work 18
  - in Link 40
  - in UOW COM class 83
- UOW COM class
  - Methods
    - BackOut 83
    - Commit 84
    - ForceReset 84
    - UowId 84
- UowId
  - in Methods
    - in UOW COM Class 84
- UserId
  - in Methods
    - in Terminal COM Class 81

- UserId
  - in Methods
    - in Connect COM Class 42
- userid (parameter)
  - in SetTermDefns 79
- userId (parameter)
  - in Details 39
- userID (parameter)
  - in Details 39
- Using BMS Map data with EPI COM classes
  - in Connecting to CICS 3270 applications using the EPI
    - in Using the COM classes 23
- Using the COM classes
  - Connecting to CICS 3270 applications using the EPI
    - CICS Server Information 23
    - EPI call synchronization types 22
    - Running a CICS 3270 session 21
    - Using BMS Map data with EPI COM classes 23
  - Making an ECI link call to CICS
    - CICS Server Information and Connection Status 17
    - ECI Call Synchronization Types 16
    - ECI Link Calls within a Unit Of Work 18

## V

- Validate
  - in Methods
    - in Map COM Class 62
- Value (parameter)
  - in SetExtAttribute 55
- VBScript 5
- VerifyPassword
  - in Methods
    - in Connect COM Class 42
    - in Terminal COM Class 81
- viewing online documentation 107
- Visual Basic 5

## W

- Wait
  - in Methods
    - in Flow COM Class 59
  - in Wait 59
- Width
  - in Methods
    - in Screen COM Class 65
- Windows 95
  - in Environments supported 5

- Windows 98
  - in Environments supported 5
- Windows NT
  - in Environments supported 5

## Y

- Year
  - in SecTime COM Class 70





Program Number: 5648-B42



Printed in the United States of America  
on recycled paper containing 10%  
recovered post-consumer fiber.

SC33-1924-01

