CICS Family

OO Programming in BASIC for CICS Clients

SC33-1924-00

CICS Family

OO Programming in BASIC for CICS Clients

SC33-1924-00

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page v.

First edition (March 1997)

This edition applies to CICS Clients Version 2.0.1, program number 5639-001, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Consult the latest edition of the applicable IBM system bibliography for current information on this product.

The previous book dealing with this subject,*Object Oriented Programming for CICS Clients* SC33-1639, has been split into two books. This book contains the BASIC programming topics. *OO Programming in C++ for CICS Clients* SC33-1923 contains the C++ programming topics.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

At the back of this publication is a page entitled “Sending your comments to IBM”. If you want to make comments, but the methods described are not available to you, please address them to:

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

When you send information to IBM, you grant IBM a nonexclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Contents

Notices

The following paragraph does not apply to any 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 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.

References in this publication to IBM products, programs, or services do not imply that IBM intends to make these available in all countries in which IBM operates. 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 of the intellectual property rights of IBM may be used instead of the IBM product, program, or service. The evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, are the responsibility of the user.

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 Laboratory Counsel, MP151, IBM United Kingdom Laboratories, 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.

IBM may have patents or pending patent applications covering subject matter 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 the IBM Director of Licensing, IBM Corporation, 500 Columbus Avenue, Thornwood, New York 10594, U.S.A.

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:

AIX	BookManager	CICS	IBM
MVS/ESA	OS/2		

Microsoft, Windows, Windows NT, the Windows 95 Logo, Visual Basic, and Visual C++ are trademarks of Microsoft Corporation.

LotusScript is a trademark of Lotus Development Corporation.

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

Preface

Application programmers can use the object oriented (OO) classes supplied with CICS Clients Version 2.0.1 to develop object oriented CICS client programs.

CICS services are available to clients through the External Call Interface (ECI) and External Programming Interface (EPI). The CICS client OLE interfaces allow a Basic 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 LotusScript.

Who this book is for

This book is for CICS application programmers who want to know how to use the OO classes provided in CICS Clients Version 2.0.1.

What this book is about

This book is divided into two parts.

Part 1 describes the programming environment and shows you, with examples, how to use the interfaces.

Part 2 contains the reference material.

Note: The previous book dealing with this subject,*Object Oriented Programming for CICS Clients* SC33-1639, has been split into two books. This book contains the BASIC programming topics. *OO Programming in C++ for CICS Clients* SC33-1923 contains the C++ programming topics.

What you need to know before reading this book

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.

Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both hardcopy and BookManager softcopy versions of a publication are in step, but subsequent updates will probably be available in softcopy before they are available in hardcopy.

For CICS Transaction Server books, these softcopy updates appear regularly on the *Transaction Processing and Data Collection Kit* CD-ROM, SK2T-0730-xx. Each reissue of the collection kit is indicated by an updated order number suffix (the -xx part). For example, collection kit SK2T-0730-06 is more up-to-date than SK2T-0730-05. The collection kit is also clearly dated on the cover.

Here's how to determine if you are looking at the most current copy of a publication:

- A publication with a higher suffix number is more recent than one with a lower suffix number. For example, the publication with order number SC33-0667-02 is more recent than the publication with order number SC33-0667-01. (Note that suffix numbers are updated as a product moves from release to release, as well as for hardcopy updates within a given release.)
- When the softcopy version of a publication is updated for a new collection kit the order number it shares with the hardcopy version does not change. Also, the date in the edition notice remains that of the original publication. To compare softcopy with hardcopy, and softcopy with softcopy (on two editions of the collection kit, for example), check the last two characters of the publication's filename. The higher the number, the more recent the publication. For example, DFHPF104 is more recent than DFHPF103. Next to the publication titles in the CD-ROM booklet and the readme files, asterisks indicate publications that are new or changed.
- Updates to the softcopy are clearly marked by revision codes (usually a "#" character) to the left of the changes.

Keeping up-to-date through the Internet

IBM CICS Development maintain a World Wide Web (WWW) home page for all members of the CICS family of products. These pages are publicly available to anyone with internet access at the following URL:

CICS Home Page URL

```
http://www.hursley.ibm.com/cics/
```

Bibliography

Here are some books that you may find useful.

CICS family: The following books are published by IBM:

- *CICS Family: General Information*, GC33-0155
- *CICS Family: Client/Server Programming*, SC33-1435
- *CICS Clients Administration*, SC33-1436
- *ITSC CICS Clients Unmasked*, GG24-2534

CICS for MVS/ESA: The following book is published by IBM:

- *CICS for MVS/ESA Application Programming Guide*, SC33-1169

CICS on Open Systems: The following book is published by IBM:

- *CICS on Open Systems Application Programming Guide*, SC33-1568

CICS for OS/2: The following book is published by IBM:

- *CICS for OS/2 Application Programming*, SC33-1585

Part 1. Client Classes—Guidance

Chapter 1. Introduction to OO programming

The CICS family provides robust transaction processing capabilities across the major hardware platforms that IBM 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 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 2.0.1 for C++ and BASIC programmers. This book covers the BASIC programming topics. If you wish to program using C++, then please refer to *OO Programming in C++ for CICS Clients* SC33-1923 where you will find a user guide based on the C++ samples and a description of the C++ classes.

Chapter 2. Establishing the working environment

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

Environments supported

Windows

Microsoft Windows Version 3.11
Microsoft Visual Basic Version 4.0
Microsoft Visual Basic Programming System, Applications Edition

Note: OLE support in the Windows 3.11 environment is for CICS ECI only, not the CICS EPI.

Windows NT

Windows NT Workstation Version 3.51
Microsoft Visual Basic Version 4.0
Microsoft Visual Basic Programming System, Applications Edition

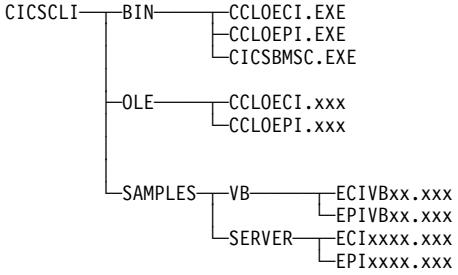
Windows 95

Windows 95
Microsoft Visual Basic Version 4.0
Microsoft Visual Basic Programming System, Applications Edition

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

Installation

OO support files for various programming languages are installed with the CICS Client. After installation, your directories will contain the following Visual Basic files:



For further information about the code samples, see the README files in the `CICSCLI\SAMPLES` directories. Additional CICS client samples are available via the IBM CICS Web page. See “Keeping up-to-date through the Internet” on page viii

Chapter 3. Using the OLE interfaces

The CICS client provides OLE interfaces for the CICS ECI and EPI on Microsoft Windows NT and Windows 95 and for CICS ECI only on Windows 3.11. The OLE interfaces are implemented as local OLE Automation Servers, CCLOECI.EXE and CCLOEPI.EXE.

These OLE interfaces can be accessed from Microsoft Visual Basic version 4.0, from Visual Basic for Applications (which is provided built-in to applications such as Microsoft Excel version 5.0.), and from Lotus Notes.

Programming Overview

Note

After installation, to use the OLE interfaces in Visual Basic programs, the ECI and EPI servers (CCLOECI.EXE and CCLOEPI.EXE) must be run once as normal Windows applications. This will register the OLE interfaces in the client operating system.

The interfaces, their methods and parameters can be viewed using the Microsoft Object Viewer, or the Object Browser provided within Visual Basic V4.0. The OLE interface information to support these tools is contained in a type library (.TLB file) shipped with the CICS client.

Note also ...

To use the type definitions and constants supplied with the CICS OLE interfaces within a Visual Basic program you must add a reference to the CICS ECI or EPI type library to your Visual Basic project. You do this using the *references* dialog in the Visual Basic *Tools* menu. The CICS type library files (CCLOECI.TLB and CCLOEPI.TLB) can be found in the \cicscli\ole directory.

Once you have added a reference to one of the CICS type libraries, you can view the CICS interfaces and constants using the *object browser* dialog in the Visual Basic *view* menu.

The interfaces that the CICS OLE servers provide are listed in the following table:

Object Linking and Embedding

Table 1. OLE ECI Interfaces		
Object	**Interface**	**Description**
Buffer	**Ccl.Buffer**	Buffer used for passing data to and from a CICS server
Connection	**Ccl.Connect**	Controls a connection to a CICS server
ECI	**Ccl.ECI**	Provides access to a list of CICS servers configured in the client
Flow	**Ccl.Flow**	Controls a single interaction with CICS server program
UOW	**Ccl.UOW**	Coordinates a recoverable set of calls to a CICS server

Table 2. OLE EPI Interfaces		
Object	**Interface**	**Description**
EPI	**Ccl.EPI**	Initializes and terminates the CICS EPI and provides access to a list of CICS servers configured in the client
Field	**Ccl.Field**	Provides access to a single 3270 field on a screen.
Map	**Ccl.Map**	Provides access to 3270 fields defined by a CICS server BMS map
Screen	**Ccl.Screen**	Provides access to a 3270 terminal screen
Session	**Ccl.Session**	Controls a sequence of 3270 terminal interactions with a CICS server
Terminal	**Ccl.Terminal**	Controls a 3270 terminal connection

Table 3. OLE Constants		
Object	**Interface**	**Description**
Constants	**Ccl.Constants**	Provides a set of constants for use with the interfaces.
Exceptions	**Ccl.Exceptions**	Provides a set of exception codes

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

Making an ECI link call to CICS

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 Object
Dim Connect As Object
Dim Flow As Object
Dim Buffer As Object
Dim UOW As Object
```

The required ECI objects are then instantiated using the Visual Basic **CreateObject** function. This can be done in the **Form_Load** subroutine or at some later stage in response to some user action. Note that a Ccl.ECI object must be created first.

```
Sub ECILink_Click()
    Set ECI = CreateObject("Ccl.ECI")
    Set Connect = CreateObject("Ccl.Connect")
    Set Flow = CreateObject("Ccl.Flow")
    Set Buffer = CreateObject("Ccl.Buffer")
```

Details of the CICS server to be used – server name (as configured in `CICSCLI.INI`) , 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, "ECIWTO", Buffer, UOW
    Text1.Text = Buffer.String
```

Finally the CICS OLE 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 OLE automation server containing the logic to process the contents of the Ccl.Buffer objects.

ECI Call Synchronization Types

The CICS client ECI OLE interfaces support synchronous (“blocking”) and deferred synchronous (“polling”) protocols. These interfaces 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 = CreateObject("Ccl.Connect")
    Set Flow = CreateObject("Ccl.Flow")
    Set Buffer = CreateObject("Ccl.Buffer")
    Connect.Details "CICSNAME", "sysad", "sysad"
    Flow.SetSyncType cclDSync
    Buffer.SetString "Hello"
    Connect.Link Flow, "ECIWTO", 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 ECIReply_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 **Ccl.ECI** interface provides the names and descriptions of CICS servers configured in the `CICSCLI.INI` file. The **Ccl.Connect** interface provides methods for querying the availability of a particular CICS server.

Object variables are declared as before, this time we use **Ccl.ECI**, **Ccl.Connect** and **Ccl.Flow** OLE interfaces:

```
'Declare object variables
Dim ECI As Object
Dim Connect As Object
Dim Flow As Object
```

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

```
Sub ECIServers_Click()
    Dim I

    'Instantiate CICS ECI objects
    Set ECI = CreateObject("Ccl.ECI")
    Set Connect = CreateObject("Ccl.Connect")
    Set Flow = CreateObject("Ccl.Flow")

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

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
```

Guide to OLE CICS ECI

ECI Link Calls within a Unit Of Work

Using the **Ccl.UOW** OLE interface, 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 = CreateObject("Ccl.Connect")
    Set Flow = CreateObject("Ccl.Flow")
    Set UOW = CreateObject("Ccl.UOW")
    Set Buffer = CreateObject("Ccl.Buffer")
    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).

Connecting to CICS 3270 applications using the EPI

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 Object
Dim Terminal As Object
Dim Session As Object
Dim Screen As Object
Dim Field As Object
```

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

The Ccl.EPI object must be created first to initialize the CICS client EPI. A Ccl.Terminal 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 CICSCLI.INI), the other parameters specify additional connection details (see the reference section “Connect” on page 51).

```
Sub EPIConnect_Click()
    'Create Ccl.EPI first to initialize 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
```

Running a CICS 3270 session

Having connected a Ccl.Terminal object to the required CICS server the **Ccl.Terminal**, **Ccl.Session**, **Ccl.Screen** and **Ccl.Field** interfaces 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 EPIStart_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
```

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 cclPF3
    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:

```
Sub EPIDone_Click()
    Term.Disconnect
    EPI.Terminate
    'Delete the EPI OLE 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 OLE interfaces support synchronous (“blocking”) and deferred synchronous (“polling”) protocols. The Visual Basic environment does not support the asynchronous calls with callback or message notification 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 = CreateObject("Ccl.Session")
    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.Poll** method is used to collect the reply from CICS:

```
Sub EPIReply_Click()
    If Terminal.Poll Then
        'Screen updated, output some fields
        Set Screen = Terminal.Screen
        Set Field = Screen.FieldByPosition(1,1)
        List1.AddItem Field.Text
    Else
        List1.AddItem "No reply from CICS yet"
    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.Poll** 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 **Ccl.EPI** interface provides the names and descriptions of CICS servers configured in the `CICSCLI.INI` 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”:

Guide to OLE CICS EPI

```
Sub EPIServers_Click()
    Dim I
    'Instantiate CICS EPI object
    Set EPI = CreateObject("Ccl.EPI")
    '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 OLE interfaces

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 OLE interface **Ccl.Map.**

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_PRODNAM = 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 **Ccl.Map** interface to access fields by their BMS symbolic names:

```
Dim EPI As Object
Dim Terminal As Object
Dim Session As Object
Dim Screen As Object
Dim Map as Object
Dim Field As Object
```

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

```
Sub EPIConnect_Click()
    'Create Ccl.EPI first to initialize 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
```

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:

```
    Map = CreateObject("Ccl.MAP")
    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 **Ccl.Screen** interface (and its **FieldByIndex** and **FieldByPosition** methods) with the use of the **Ccl.Map** interface.

Guide to OLE CICS EPI

Part 2. OLE Interfaces — Reference

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

Chapter 4. Ccl.Buffer interface

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

The Ccl.Buffer 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 Ccl.Buffer object to be accessed as a Byte array. The Ccl.Buffer 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 OLE interface converts this to and from single-byte ANSI.

When a **Ccl.Buffer** 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.

Methods

AppendString

AppendString

AppendString(*string* **As String)**

string The source string.

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

Data

Data

Data() As Variant

Returns the contents of the buffer as a Byte array.

OLE Interface: Ccl.Buffer

ExtractString

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

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

Length() As Integer

Returns the length of the data area in bytes.

Overlay

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

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

SetLength(*length* **As Integer)**

length The new length of the data area, in bytes

Changes the current length of the data area.

SetString

SetString

SetString(*string* **As String)**

string Source string

Copies the supplied string into the object.

String

String

String() As String

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

OLE Interface: Ccl.Buffer

Chapter 5. Ccl.Connect interface

The **Ccl.Connect** interface 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 **Ccl.Connect** interface 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 **Ccl.Flow** object and a **Ccl.Connect** object.

Methods

Cancel

Cancel

Cancel(*flow* **As Object)**

flow The **Ccl.Flow** 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

Changed(*flow* **As Object)**

flow The **Ccl.Flow** 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.

OLE Interface: Ccl.Connect

Details

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 `CICSCLI.INI`—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

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

flow	The **Ccl.Flow** 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.
commArea	A **Ccl.Buffer** 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 **Ccl.UOW** 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.

ServerName

ServerName

ServerName() As String

Returns the name of the server system held by the **Ccl.Connect** object and listed by `CICSCLI.INI`, or blanks if the default server is being used and no calls have yet been made.

ServerStatus

ServerStatus

ServerStatus() As Integer

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

`cclStatUnknown`	The CICS server status is unknown
`cclStatAvail`	The CICS server is available
`cclStatUnavail`	The CICS server is not available

Constant definitions for these codes are provided in `CCLOECI.TLB`.

ServerStatusText

ServerStatus Text

ServerStatusText() As String

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

Status

Status

Status(*flow* **As Object)**

flow	The **Ccl.Flow** object used to control the client/server call

Request the status of the server connection.

OLE Interface: Ccl.Connect

TranDetails

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 program are attached. The default is to use the default CPMI. 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.

UserId

UserId

UserId() As String

Returns the user ID held by the **Ccl.Connect** object, or blanks if none.

Chapter 6. Ccl.ECI Interface

All applications using the ECI OLE interface must first create a Ccl.ECI object.

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

Methods

ErrorWindow

ErrorWindow

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

ExCode

ExCode() As Integer

Returns an enumeration that indicates the last ECI error. Constant definitions for the possible values are provided in `CCL0ECI.TLB`. Note that the **ExCodeText** method returns a descriptive text string for the error value.

ExCodeText

ExCodeText

ExCodeText() As String

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

OLE Interface: Ccl.ECI

ServerCount

ServerCount

ServerCount() As Integer

Returns the number of candidate servers to which the client may be connected, as configured in the `CICSCLI.INI` file.

ServerDesc

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

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.

Chapter 7. Ccl.EPI Interface

The **Ccl.EPI** interface 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 Ccl.EPI object before you create **Ccl.Terminal** objects to connect to CICS servers. The **Diagnose**, **ExCode**, and **State** methods provide information on error conditions.

Methods

Diagnose

Diagnose

Diagnose() As String

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

ErrorWindow

ErrorWindow

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

ExCode

ExCode() As Integer

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.

Constant definitions for these codes are provided in `CCLOEPI.TLB`.

OLE Interface: Ccl.EPI

ServerCount

ServerCount

ServerCount() As Integer

Returns the number of candidate servers to which the client may be connected, as configured in the `CICSCLI.INI` file.

ServerDesc

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 `CICSCLI.INI` file for the specified server.

ServerName

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 `CICSCLI.INI` file for the specified server.

State

State

State() As Integer

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

`cclActive`	Initialized
`cclDiscon`	Terminated
`cclError`	Error, call **ExCode** and **Diagnose** methods for further information.

Constant definitions for these codes are provided in `CCLOEPI.TLB`.

Terminate

Terminate

Terminate()

Terminates the CICS client EPI in a controlled manner.

OLE Interface: Ccl.EPI

Chapter 8. Ccl.Field Interface

The **Ccl.Field** interface is used to access a single field on a 3270 screen. **Ccl.Field** objects are created and deleted when 3270 data from the CICS server is processed by a **Ccl.Screen** object.

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

Methods

AppendText

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

BackgroundColor() As Integer

Returns a value which indicates the background color of the field. Constant definitions for these values (`cclBlue`, `cclRed`, etc) are supplied in `CCLOEPI.TLB` and are listed in "Ccl.Field Color Attribute Values" on page 60.

Column

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.

OLE Interface: Ccl.Field

DataTag

DataTag

DataTag() As Integer

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

```
cclModified
cclUnmodified
```

Constant definitions for these values are supplied in `CCLOEPI.TLB`.

ForegroundColor

ForegroundColor

ForegroundColor() As Integer

Returns a value which indicates the foreground color of the field. Constant definitions for these values are supplied in `CCLOEPI.TLB` and are listed in “Ccl.Field Color Attribute Values” on page 60.

Highlight

Highlight

Highlight() As Integer

Returns a value which indicates which type of highlight is being used. Constant definitions for these values are supplied in `CCLOEPI.TLB` and are listed in “Ccl.Field Highlight Attribute Values” on page 60.

InputProt

InputProt

InputProt() As Integer

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

```
cclProtect
cclUnprotect
```

InputType

InputType

InputType() As Integer

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

```
cclAlphanumeric
cclNumeric
```

Intensity

Intensity

Intensity() As Integer

Returns a value which indicates whether the field is normal, intense or dark. Possible values are:

```
cclDark
cclNormal
cclIntense
```

Length

Length

Length() As Integer

Returns the length of the field.

Position

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.

OLE Interface: Ccl.Field

ResetDataTag

ResetDataTag

ResetDataTag()

Resets the modified data tag (MDT) to `cclUnmodified`.

Row

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.

SetText

SetText

SetText(*textString* **As String)**

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

Copies *textString* into the field.

Text

Text

Text() As String

Returns the text currently held in the field.

TextLength

TextLength

TextLength() As Integer

Returns the number of characters currently held in the field.

Transparency

Transparency

Transparency() As Integer

Returns a value which indicates the background transparency of the field. Constant definitions for these values are supplied in `CCLOEPI.TLB` and are listed in “Ccl.Field Transparency Attribute Values” on page 60.

OLE Interface: Ccl.Field

Chapter 9. Ccl.Flow interface

A **Ccl.Flow** object is used to control ECI communications for a client/server pair.

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

Methods

AbendCode

AbendCode

AbendCode() As String

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

Poll

Poll

Poll(*commArea* **As Object) As Boolean**

commArea A **Ccl.Buffer** 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

SetSyncType(*syncType* **As Integer)**

syncType The synchronization type required for this **Ccl.Flow** object. Possible values are:

`cclSync`

`cclDSync`

Sets the synchronization type required for this **Ccl.Flow** 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.

Wait

Wait

Wait()

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. Ccl.Map interface

The **Ccl.Map** interface 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.

Methods

ExCode

ExCode

ExCode() As Integer

Returns a value which indicates the current condition code. Constant definitions are supplied in `CCLOEPI.TLB`.

FieldByName

FieldByName

FieldByName(*name* **As Integer) As Object**

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 **Ccl.Field** object.

Validate

Validate

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

screenRef **Ccl.Screen** 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. Ccl.Screen Interface

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

A single **Ccl.Screen** object is created by the **Ccl.Terminal** object when the **Ccl.Terminal.Connect** method is used to establish a connection to a CICS server. The application gets access to the **Ccl.Screen** object via the **Ccl.Terminal.Screen** method.

Methods

CursorCol

CursorCol

CursorCol() As Integer

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

CursorRow

CursorRow

CursorRow() As Integer

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

Depth

Depth

Depth() As Integer

Returns the number of rows on the screen.

OLE Interface: Ccl.Screen

FieldByIndex

FieldByIndex

FieldByIndex(*index* **As Integer) As Object**

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

FieldByPosition

FieldByPosition

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

rowPos The row number of the field (topmost row = 1).
colPos The column number of the field (leftmost column = 1).

FieldCount

FieldCount

FieldCount() As Integer

Returns the number of fields on the screen.

SetAID

SetAID

SetAID(*key* **As Integer)**

key The AID key value. Constant definitions for these values are supplied in `CCLOEPI.TLB` and are listed in “Ccl.Screen AID key codes” on page 60.

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

SetCursor

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

Width() As Integer

Returns the number of columns on the screen.

OLE Interface: Ccl.Screen

Chapter 12. Ccl.Session interface

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

Methods

SetSyncType

SetSyncType

SetSyncType(*syncType* **As Integer)**

syncType The synchronization type required for this **Ccl.Session** object. Possible values are:

`cclSync`
`cclDSync`

Sets the synchronization type required for this **Ccl.Session** 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

State() As Integer

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.

Constant definitions for these values are supplied in `CCLOEPI.TLB`.

OLE Interface: Ccl.Session

Chapter 13. Ccl.Terminal interface

The **Ccl.Terminal** interface 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.

Methods

Connect

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 `CICSCLI.INI`, 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.

Diagnose

Diagnose

Diagnose() As String

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

OLE Interface: Ccl.Terminal

Disconnect

Disconnect

Disconnect()

Disconnect terminal from CICS.

ExCode

ExCode

ExCode() As Integer

Returns a value which indicates the most recent condition code returned by the server. Constant definitions for these values are supplied in `CCLOEPI.TLB`.

NetName

NetName

NetName() As String

Returns the network name of the terminal.

Poll

Poll

Poll() As Boolean

Indicates whether a reply has been received from a deferred synchronous **Start** or **Send** request. Possible values are:

True	A reply has been received
False	A reply has not 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.Poll** 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`.

QueryATI

QueryATI

QueryATI() As Integer

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

```
cclATIEnabled
cclATIDisabled
```

Screen

Screen

Screen() As Object

Returns the **Ccl.Screen** object that is handling the 3270 screen associated with this terminal.

Send

Send

Send(*session* **As Object)**

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

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

ServerName

ServerName

ServerName() As String

Returns the name of the server system held by the **Ccl.Terminal** object and listed by `CICSCLI.INI`, or blanks if the default server is being used and no calls have yet been made.

OLE Interface: Ccl.Terminal

SetATI

SetATI

SetATI(*stateVal* **As Integer)**

stateVal A value which indicates whether the ATI is to be enabled or disabled. Possible values are:

`cclATIEnabled`
`cclATIDisabled`

Start

Start

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

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

State() As Integer

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 **cclSession** interface. They 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.

Constant definitions for these values are supplied in `CCLOEPI.TLB`.

TransId

TransId

TransId() As String

Returns the 4-character name of the current CICS transaction.

OLE Interface: Ccl.Terminal

Chapter 14. Ccl.UOW interface

Use this interface 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 **Ccl.UOW** object — see **Link** method in **Ccl.Connect** (“Link” on page 26).

Methods

BackOut

BackOut

BackOut(*flow* **As Object)**

flow The **Ccl.Flow** 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

Commit(*flow* **As Object)**

flow The **Ccl.Flow** object which is used to control the client-server call

Terminate this UOW and commit all changes made to recoverable resources in the server.

OLE Interface: Ccl.UOW

Chapter 15. OLE Global Constants

Constants and interface definitions for the CICS Client OLE interfaces are provided in two OLE Type Libraries `CCLOECI.TLB` and `CCLOEPI.TLB`, which are installed in the `\cicscli\ole` directory.

For compatibility with Visual Basic programs written for the ECI OLE support provided in CICS Clients Version 2.0.1, a set of constants is also supplied in a Visual Basic file `CCLOECI.BAS`.

CICS Client ECI Constants

These constants are defined in file `CCLOECI.TLB`.

Synchronization types

`cclSync`	synchronous call type
`cclDSync`	deferred synchronous call type

Ccl.Status connection and status codes

`cclStatUnknown`	The CICS server status is unknown
`cclStatAvail`	The CICS server is available
`cclStatUnavail`	The CICS server is not available

CICS Client EPI Specific Constants

These constants are defined in file `CCLOEPI.TLB`.

Synchronization types

`cclSync`	synchronous call type
`cclDSync`	deferred synchronous call type

Ccl.EPI/Ccl.Terminal states

`cclActive`	**Ccl.EPI** initialized/**Ccl.Terminal** connected
`cclServer*`	Transaction in progress in the CICS server.
`cclClient*`	CICS server is waiting for a response from the client
`cclDiscon`	**Ccl.EPI** terminated/**Ccl.Terminal** disconnected
`cclError`	**Ccl.EPI**/**Ccl.Terminal** error, call **ExCode** and **Diagnose** methods for further information.

* **Ccl.Terminal** only

OLE Global Constants

Ccl.Terminal ATI states

cclATIEnabled	Automatic Transaction Initiation enabled
cclATIDisabled	Automatic Transaction Initiation disabled

Ccl.Screen AID key codes

cclEnter	Enter key
cclClear	Clear key
cclPA1 thru cclPA3	Program attention keys
cclPF1 thru cclPF24	Program function keys

Ccl.Field 3270 Base Attribute Values

cclProtect	Protected field
cclUnprotect	Unprotected (input) field
cclAlphanumeric	Alphanumeric input field
cclNumeric	Numeric input field
cclNormal	Normal display
cclIntense	Intensified display
cclDark	Non-display field
cclUnmodified	Field has not been modified
cclModified	Field has been modified

Ccl.Field Color Attribute Values

cclDefaultColor	cclBlue	cclRed	cclPink
cclGreen	cclCyan	cclYellow	cclNeutral
cclBlack	cclDarkBlue	cclOrange	cclPurple
cclPaleGreen	cclPaleCyan	cclGray	cclWhite

Ccl.Field Highlight Attribute Values

cclHltDefault	Default field text highlighting
cclHltNormal	Field text highlight as specified by 3270 base attribute
cclHltBlink	Blinking text
cclHltReverse	Reverse video text
cclHltUnderscore	Underscored text
cclHltIntense	High intensity text

Ccl.Field Transparency Attribute Values

cclTrnDefault	Default (opaque) field background
cclTrnOr	Transparent field background (OR)
cclTrnXor	Transparent field background (XOR)
cclTrnOpaque	Opaque field background

CICS Client ECI/EPI Exceptions

These constants are defined in type libraries CCLOECI.TLB and CCLOEPI.TLB.

These constants are defined in file CCLOECI.BAS.

Exception codes

cclNoError	currently, there is no error to report
cclBufferOverflow	a fixed-length Buffer object overflowed
cclMultipleInstance	attempted to construct more than one **Ccl.ECI** or **Ccl.EPI** object or subclass instance
cclActiveFlow	attempted an invalid operation on an active Ccl.Flow object
cclActiveUOW	attempted an invalid operation on an active Ccl.UOW object
cclSyncType	used a CclFlow synchronization type incorrectly
cclThreadCreate	an error occurred during thread creation
cclThreadWait	an error occurred while waiting for a thread to terminate
cclThreadKill	an error occurred while attempting to terminate a thread
cclDataLength	the length of a COMMAREA Buffer on a CclConn::Link call was out of range
cclNoCICS	the CICS client or server was not available
cclCICSDied	the CICS server became unavailable while a UOW was active
cclNoReply	no reply was available from the server
cclTransaction	the CICS transaction that executed the requested program abended
cclSystemError	an internal CICS client system error occurred.
cclResource	the client or server had insufficient resources to complete the request
cclMaxUOWs	exceeded the maximum number of UOW's the configuration will support
cclUnknownServer	could not locate the requested server
cclSecurity	did not supply a valid userId/password for the server
cclMaxServers	exceeded the maximum number of servers the configuration will support
cclMaxRequests	exceeded the maximum number of simultaneous server requests the configuration will support
cclRolledBack	the server could not commit the changes in a UOW, so backed them out instead
cclParameter	input parameter invalid
cclInvalidState	the object is in an invalid state for the requested operation
cclTransId	invalid transaction identifier
cclInitEPI	EPI not initialized
cclConnect	unable to connect to requested CICS server
cclDataStream	unsupported 3270 datastream received from CICS
cclInvalidMap	BMS map object does not match current screen
cclClass	an internal error occurred in the Client Classes

OLE Global Constants

Glossary

AID. Attention Identifier

ATI. Automatic Transaction Initiation

BMS. Basic Mapping Support

COMMAREA. CICS Communications Area

ECI. External Call Interface

EPI. External Presentation Interface

ISC. Inter-Systems Communication

OLE. Object Linking and Embedding

UOW. Unit of Work

Index

D

E

F

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

Sending your comments to IBM

CICS Family

OO Programming in BASIC
for CICS Clients

SC33-1924-00

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book. Please limit your comments to the information in this book and the way in which the information is presented.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:

- By mail, use the Readers' Comment Form (RCF)
- By fax:
  - From outside the U.K., after your international access code use 44 1962 870229
  - From within the U.K., use 01962 870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: WINVMD(IDRCF)
  - Internet: idrcf@winvmd.vnet.ibm.com

Whichever you use, ensure that you include:

- The publication number and title
- The page number or topic to which your comment applies
- Your name/address/telephone number/fax number/network ID.

Readers' Comments

CICS Family

OO Programming in BASIC
for CICS Clients

SC33-1924-00

Use this form to tell us what you think about this manual. If you have found errors in it, or if you want to express your opinion about it (such as organization, subject matter, appearance) or make suggestions for improvement, this is the form to use.

To request additional publications, or to ask questions or make comments about the functions of IBM products or systems, you should talk to your IBM representative or to your IBM authorized remarketer. This form is provided for comments about the information in this manual and the way it is presented.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate without incurring any obligation to you.

Be sure to print your name and address below if you would like a reply.

Name	Address
Company or Organization	
Telephone	Email

CICS Family
OO Programming in BASIC for CICS Clients SC33-1924-00

You can send your comments POST FREE on this form from any one of these countries:

Australia	Finland	Iceland	Netherlands	Singapore	United States
Belgium	France	Israel	New Zealand	Spain	of America
Bermuda	Germany	Italy	Norway	Sweden	
Cyprus	Greece	Luxembourg	Portugal	Switzerland	
Denmark	Hong Kong	Monaco	Republic of Ireland	United Arab Emirates	

If your country is not listed here, your local IBM representative will be pleased to forward your comments to us. Or you can pay the postage and send the form direct to IBM (this includes mailing in the U.K.).

1 Cut along this line

2 Fold along this line

By air mail
Par avion

IBRS/CCRI NUMBER: PHQ - D/1348/SO

NE PAS AFFRANCHIR

NO STAMP REQUIRED

REPONSE PAYEE
GRANDE-BRETAGNE

IBM United Kingdom Laboratories
Information Development Department (MP095)
Hursley Park,
WINCHESTER, Hants
SO21 2ZZ United Kingdom

3 Fold along this line

From: Name _____

Company or Organization _____

Address _____

_____

EMAIL _____

Telephone _____

4 Fasten here with adhesive tape

1 Cut along this line

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

SC33-1924-00

Spine information:

CICS Family

OO Programming in BASIC for CICS Clients