Changing existing CICS 3270 applications to enable new Client/Server and Web access

Version 1.2

Document: CA03 SCRIPT
Issued: 17th March, 1998
Revision Date: 18th January, 1999
Previous Revision Date: 18th May, 1998
Next Review: As required

Jim MacNair
IBM Transaction Systems
MS 1272
Route 100
Somers, NY 10589
macnair@us.ibm.com

CICS
Application Server

Take Note!

Before using this User's Guide and the product it supports, be sure to read the general information under "Notices".

First Edition, March 1998

This edition applies to Version 1.1 of Changing existing CICS 3270 applications to enable new Client/Server and Web access and to all subsequent releases and modifications until otherwise indicated in new editions.

A form for reader's comments is provided at the back of this publication. If the form has been removed, address your comments to:

IBM United Kingdom Laboratories
Transaction Systems Marketing Support (MP207)
Hursley Park
Hursley
Hampshire, SO21 2JN, England

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you. You may continue to use the information that you supply.

Contents

Notices.

The following paragraph does not apply in 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 licensed program or other IBM product in this publication is not intended to state or imply that only IBM's program or other product may be used. Any functionally equivalent program that does not infringe any of the intellectual property rights may be used instead of the IBM product. Evaluation and verification of operation in conjunction with other products, except those expressly designated by IBM, is the user's responsibility.

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, USA.

The information contained in this document has not be submitted to any formal IBM test and is distributed AS IS. The use of the information or the implementation of any of these techniques is a customer responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms are trademarks of the International Business Machines Corporation in the United States and/or other countries:

CICS

CICS/ESA

IBM

IBM Transaction Server for Windows NT

OS/2

VisualAge

AIX

The following terms are trademarks of the Microsoft Corporation in the United States and/or other countries:

Windows

Windows NT

Microsoft Windows NT Advanced Server

Microsoft Windows NT Workstation

Microsoft Visual Basic

Microsoft Visual C++

The following terms are trademarks of other companies:

PowerBuilder, Sybase Corporation

Java, Sun Microsystems, Inc.

Acknowledgments

The DFHCBRW.CPP sample program from the IBM Transaction Server for Windows NT has been changed to operate in a pseudo-conversational manner rather then conversationally. This modified program is supplied with this SupportPac. It is copyrighted by IBM Corporation.

Summary of Changes

Date	**Changes**
17th March, 1998	Initial release
12th November, 1998	Release 1.1
18th January, 1999	Release 1.2

The following changes were made in this release 1.2:

- Fixed bug with ERASESW invalid character
- Changed handling of SEND parameters to fix problems
- Added support for -INC statements (includes)
- Added support to handle ASSIGN TERMCODE
- Fixed bug with COPY at end of variable name
- Fixed bug with END PROGRAM and generation of BADMAPNAME paragraph
- Write message for LINK and XCTL statements
- set USRCALEN to 0 on simple EXEC CICS RETURN
- Fixed bugs in sample browse program

The following changes were made in this release 1.1:

- Added support for COBOL copy books
- Added statistics/counters
- Added support for ALARM keyword on EXEC CICS SEND
- Fixed following bugs
  - Multiple keywords in line
  - Parameter string in variable name

Bibliography

- *CICS/ESA Application Programmer's Reference, IBM Corp. SC33-1170*

Introduction

Objective

The objective of this SupportPac is to provide a methodology to change existing 3270 based CICS COBOL applications to allow access from a variety of client environments, while preserving a 3270 front end. Worked samples and supporting utilities are included.

Most existing CICS programs were written for 3270 terminals, using BMS maps. These programs have many dependencies on 3270 terminals, some direct and obvious and others that are more subtle. The target for this offering is existing 3270-based CICS COBOL applications. The conversion technique described is intended to allow these programs to be enabled for new client front ends at minimum cost, while preserving a 3270 front end.

This SupportPac is focused on CICS server applications. The sample programs used here are also used in a set of companion SupportPacs, which focus on client applications. Each companion client SupportPac demonstrates a different client environment accessing the sample programs provided with this SupportPac. The combination provides a complete solution.

SupportPac Overview

This SupportPac consists of three parts, namely:

1. Process to convert existing 3270 CICS COBOL applications to allow them to be called by CICS Client External Call Interface (ECI) applications.
2. Worked samples, using two COBOL programs from the IBM Transaction Server for Windows NT™ Installation Verification Programs.
3. Utility programs to assist with the conversion.

It is assumed that the reader understands CICS application programming in COBOL. While some explanations of key CICS programming concepts and features are included, this document has been written for CICS application programmers, and is fairly detailed and technical.

Software environment and prerequisites

This SupportPac was developed for the 32-bit windows environment, using Microsoft Windows NT™. The CICS programs and conversion techniques have no dependencies on Windows NT, and are aimed at a CICS/ESA environment, The conversion utilities are written for and require a 32-bit windows environment. They have only been tested with Windows NT V4.0 but should operate under Windows 95 and Windows 98.

This SupportPac was developed using Microsoft Windows NT Advanced Server V4.0™, and should work equally well on the Workstation version of Windows NT. In addition, the following additional products were used:

- IBM Transaction Server for Windows NT V4.01™
- IBM Visualage for COBOL V2.1™ (Standard edition)

The CICS clients are included with the Transaction Server product.

The utility programs are written in C, and were compiled with Microsoft Visual C++ V5 (and V4.2 as well). Source as well as executable versions are provided, although no C libraries are supplied with this SupportPac. The supplied source code allows tailoring of the utilities for individual environments. However, if any changes are made, then a copy of Microsoft Visual C++™ (V4 or V5) is required.g

IBM CICS Clients overview

CICS clients support access from non-CICS environments to CICS applications. The clients provide three key functions, two programming interfaces and a terminal emulator function.

The two programming interfaces are the External Call Interface (ECI) and the External Programming Interface (EPI). The ECI uses a Call/Return model, and is a CICS specific form of remote procedure call. The EPI is a 3270 oriented protocol, to support *screen scraping* applications.

IBM offers CICS clients on a variety of popular Client platforms, including most forms of Microsoft™ Windows (Windows 3.11, Windows 95, Windows NT, etc), OS/2, Apple MacIntosh™, PC/DOS, etc. The clients support a number of protocols to connect to CICS server systems, with the most popular being TCP/IP and SNA. The clients can connect to multiple CICS server systems at the same time.

The clients are small and efficient. They do not require a full CICS environment on the client system. They must be connected to one or more CICS servers. CICS servers are available on a variety of platforms, including smaller server systems running OS/2 or Microsoft Windows NT™, mid range systems running popular versions of Unix and IBM mainframe environments (MVS and VSE).

CICS External Call Interface (ECI) Overview

The CICS External Call Interface (ECI) is one of the two CICS client programming interfaces. It allows a program running outside of the CICS environment to access CICS programs by calling them as subroutines. The ECI includes support for CICS security, units of work, etc. Unlike most RPC environments, it does not require stubs, interface definition language (idl) compilers, etc, and it therefore quite easy to use.

CICS ECI Programming Overview

A CICS application which is called from a CICS client using the ECI can be any CICS program which can be called as a subroutine with EXEC CICS LINK, and which does not depend on a terminal. The CICS application receives a user COMMAREA as input and returns the updated COMMAREA to the calling application.

An ECI program must build two data areas and then issue the actual ECI call. The CICS client sends the request to the CICS server. The server receives the request, performs any necessary data translation and starts a transaction to process the request. When the transaction is complete, the CICS server will return the updated COMMAREA to the requesting client. The ECI call will generate a return code, to indicate any errors. The return code should be checked, and if the call was successful, the updated COMMAREA can then be used.

The ECI supports a variety of options, including synchronous and asynchronous (non-blocking) invocations, multiple calls within a single unit of work, user defined transaction codes, and other options. The programs in this SupportPac use simple synchronous calls.

Using the ECI

The COMMAREA contains all user data to be passed to the CICS application. The CICS application must place any data it is returning to the ECI application in the COMMAREA, and the updated COMMAREA is returned to the ECI application. COMMAREAs can be up to 32,500 bytes long. Larger data areas must be segmented.

The second data area that the ECI program must build is the ECIBLOCK. This area contains the parameters for the ECI call. For example, it contains the name of the CICS program to be executed, the CICS user id and password to be used, the address and length of the COMMAREA and the type of call.

The ECI area should be initialized to binary zeros. The necessary fields should be filled in, and the ECI call issued. The return code should be checked. For normal return codes, the COMMAREA should have been updated by the CICS application, and no information is returned in the ECIBLOCK. For two exceptional return codes, useful debugging data are returned in the ECIBLOCK. For a sample of a typical ECI call, see the performECI routine in the ECIMSG.C program.

CICS Client operating environment

The CICS client requires a separate daemon process on the client system. The daemon process has no user interface. This process can be started from a command line, as a Windows NT service or will be started automatically when the first client call is issued. Once started the client process remains running. It can be shut down from a command line program. The client offers powerful trace and debugging options, which can be turned on and off with a command line utility. The client uses a client definition file (CICSCLI.INI) to specify many configuration options.

Asynchronous calls

ECI calls can be either synchronous (blocking) or asynchronous (non-blocking). For asynchronous calls, there are several ways for the calling application to receive notification that a call has completed, including the use of a semaphore, a windows message or a call back routine. The application receives two return codes. When the call is first made, the CICS client will validate the call parameters, and if no problem is found, will initiate the request and return a normal return code. If a problem is found, then the return code will be set to reflect the error and no attempt is made to send the request to the CICS server. When the request is complete, the application will receive a second return code, again indicating if the call was successful or not.

Multiple calls in the same unit of work

The CICS clients support combining more than one ECI call into a single transaction. If a later call in the same unit of work fails, then the entire transaction will be rolled back, including the effects of any earlier calls in the same unit of work. This facility is known as extended units of work.

CICS Programs Called by the ECI

A CICS program which is called from a CICS client using the ECI can be any CICS subroutine which can be called with an EXEC CICS LINK statement, and which has no dependencies on a 3270 terminal. The CICS program must get its input from a CICS COMMAREA and use the COMMAREA to return any data.

CICS programs called from ECI clients do not have a 3270 terminal as a principal facility. They cannot use any terminal or Basic Mapping (BMS) facilities, and should not use any fields in the Execute Interface Block (EIB) related to terminals. In particular, they cannot:

- Use BMS calls, such EXEC CICS RECEIVE MAP

- Use EXEC CICS SEND TEXT to send messages to the terminal
- Refer to fields within BMS maps (data, attributes, length, etc)
- Refer to terminal associated fields
  - 4 character terminal name (EIBTRMID)
  - the AID byte (EIBAID or use of EXEC CICS HANDLE AID)
  - cursor position (EIBCPOSN)
  - operator id and/or initials
- Use a terminal user area (EXEC CICS ADDRESS TCTUA)
- Use EXEC CICS RETURN with either the TRANSID and/or COMMAREA and LENGTH options

In the advanced topics section, some other topics are discussed, which do not explicitly appear in a program but which can be significant considerations. This includes security, data translation, user transaction codes, etc.

Since many existing CICS programs use some of the above facilities, techniques are needed to replace the above facilities with a minimum of change to the existing applications.

Conversion options

There are two techniques commonly used to access existing production systems from a new client front end, changing the CICS programs to allow them to be called as subroutines, or having the client program pretend to be a 3270 terminal. This technique is often referred to as *screen scraping*.

Most programmers start with the assumption that they cannot change the existing back end programs. This results in the clients being programmed to appear as a 3270 terminal to the back end programs.

ECI versus "Screen Scraping" tradeoffs

There are two main techniques used to access existing production systems from a new client front end. In so-called *screen scraping*, the client application is programmed to appear as a 3270 terminal to the back end programs. With the second technique, the CICS programs are changed to allow them to be called directly by the client.

CICS clients offer a 3270 interface, known as the External Presentation Interface (EPI), which can use existing 3270 programs without requiring any change. This is attractive, and there are cases where this is the best approach (or possibly the only approach), such as cases where purchased packages are involved, etc.

The advantage of using the EPI is that no changes are required to the back end (CICS) programs. However, this may not be the best solution in many cases. Programmers who use screen scraping techniques tend to encounter the following problems:

- Complex, inefficient and error prone programming environment
- Little or no reuse between different client environments
- Conflict with the stateless nature of the Web environment
- Cannot combine multiple host interactions into a single transaction

Complex and Error Prone Programming Environment

The 3270 terminal was designed for human viewing. It was designed in a time when many users (typically 10-30) shared a single slow (1200-2400 BPS) telephone line to access a central computer. The 3270 data stream was designed to minimize the amount of data that was transmitted, and offered no programming capabilities on the terminal (client) side. Host CICS programmers are shielded from the complexities of the 3270 data stream by the CICS Basic Mapping Support (BMS), which was designed for application programming. Only the terminal hardware had to deal with the actual 3270 data stream.

When a PC program tries to simulate the 3270 terminal, it must deal with the complicated 3270 data stream arriving at the PC, and must return a similarly complex data stream to CICS. Interpreting 3270 data streams can be difficult, tedious and error prone. While it saves effort on the host side, the cost on the client side can be substantial and wipe out the apparent savings.

To further complicate matters, unsolicited or unexpected data streams can arrive instead of the usual application generated data stream. For example, CICS operator broadcasts or CICS generated error messages are expected to notify a human being sitting at a terminal of some situation. When arriving at a program, they can cause problems for the receiving program. This can cause screen scraping applications to be complex and unreliable.

Some examples of exceptional situations would include:

- Transaction abends
- Application errors (for example, File not available, Record changed by other user, etc)
- CICS operator broadcasts (for example, System shutting down in 10 minutes)

It is very easy for an EPI application to get confused in these sorts of situations, and to behave rather unpredictably.

Reuse with Multiple client environments

A key advantage of converting CICS programs to be callable via the ECI is that these transactions can be used by multiple front ends. For example, it is common for customers to want to access CICS back end transactions from many different environments, such as:

1. Customer written GUI front end, often using multiple client environments, such as:
   - HTML and CGI program using C
   - Microsoft Visual Basic™
   - GUI application using C++
   - PowerBuilder™
   - Numerous other languages and/or tools
2. Web applications such as JAVA applets or servlets
3. MQS (IBM MQSeries) applications

If a screen scraping approach is used, the client handling of the CICS 3270 data streams must be redeveloped for each additional front end environment.

Stateless nature of the Web

CICS 3270 applications expect to have a persistent connection with their terminal, and expect the server side (for example, CICS) to control the transaction flow. They assume that the user signs on once to the terminal, and then runs many transactions.

The web is by design stateless, and the transaction flow is driven by the client browser. Currently, web applications use a *stateless* approach, and do not support persistent connections.

In a typical 3270 usage scenario, a terminal is installed in the beginning of a session, and the user then signs on to this terminal, and continues to use the same terminal for the duration of the session. Transaction context is saved in CICS, and is associated with the terminal. Therefore, to effectively drive 3270 transactions from a client, the client must be able to maintain a persistent connection to CICS. However, in the case of the web, clients are stateless and cannot maintain persistent connections. When a CGI program is run, it is run in a transient process, which is shut down after each interaction with the user.

While solutions are technically possibly, they usually involve elaborate structures involving daemon processes which persist on a server, and maintain the connections, with shared memory and interprocess communications to communicate with the web applications. Timeouts are also necessary, since there is no indication if a client has terminated its interactions with a particular system and has no plans to return. Further discussion of such a design is beyond the scope of this SupportPac. Rather, the reader should assume that such a solution is quite complex, and should not be considered as normal application programming. This sort of programming is best left to vendors.

Multiple host interactions in a single transaction

In some cases, it is desirable to integrate more than one existing CICS application into a single new transaction. When a client application drives a host CICS transaction using a 3270 data stream, each host interaction must be a separate transaction. The client has no control over CICS units of work, and cannot combine the different transactions into a single unit of work. The ECI supports making multiple calls as a single unit of work. This facility is called extended units of work.

Conversion of CICS 3270 Applications to ECI Applications

This section describes the changes that must be made to an existing 3270 based CICS application to allow it to be called from a client application using the ECI. Again, the objective of this technique is to offer a least cost approach to such conversions, so every attempt is made to minimize the changes to the program, especially to program logic.

CICS BMS programs

A typical CICS BMS application will start off with an EXEC CICS RECEIVE MAP command, to retrieve user input, and will end with a EXEC CICS SEND MAP command, followed shortly by a return. Thus, the basic logic is:

```
Begin Program
EXEC CICS RECEIVE MAP
Program logic
EXEC CICS SEND MAP
EXEC CICS RETURN
End of program
```

Dependencies on 3270 Terminals

CICS applications written for 3270 terminals typically have many areas where they depend on 3270 terminal facilities. They typically include some of the following:

- BMS Map field names, lengths and attributes
- CICS BMS Commands
- SEND TEXT or Terminal Commands
- TCT User Area
- Return, including Transid and/or COMMAREA options
- Host initiated transactions
- Use of certain terminal related fields in the EIB, including:
    - Terminal Name (EIBTRMID)
    - Transaction Name (EIBTRNID)
    - Cursor Position (EIBCPOSN)
    - Aid Byte (EIBAID)
    - COMMAREA Length (EIBCALEN)
- Menu programs and RETURN IMMEDIATE
- HANDLE AID and or use of the EIBAID field

Application Conversion Process Overview

It is recommended that the following steps be followed to convert an existing 3270 based CICS application so that it can be called by ECI client applications.

- Gather the program and BMS source files, and review them
- Create a new CICS COMMAREA to be passed to the converted program
- Convert the CICS program (a utility is provided as part of this package which can automate much of the conversion)
- Compile the new program and add the necessary CICS definitions, includes a PPT entry and DFHCNV entry for the new CICS COMMAREA
- Test the new program
- If desired, create a new 3270 front end program to allow the converted program to support 3270 users

The rest of this chapter provides details on each of the steps listed, especially on designing the new CICS COMMAREA and converting the application program.

Conversion Process details

The steps in the conversion process will now be covered in detail.

Gather the Programs and BMS maps

As the first step in any application development process, it is recommended that some time be spent to understand the objectives, and the current applications. It is strongly recommended that this include actually running the transactions to be converted on a 3270 terminal or terminal emulator, and understanding how they work from a user perspective, and should include any sign on and other required actions.

The source files for the programs and BMS maps must be located, including any related copy books. It is a good idea to locate any available documentation as well. While the applications do not need to be understood in detail, it is a good idea to at least glance at the source code.

Design new CICS COMMAREA

The first step in converting an existing CICS 3270 application is to design a new CICS COMMAREA. The design should minimize the changes to the program. The following basic structure is recommended:

- Standard header, including text area
- Terminal Control Table User Area (TCTUA), if used in the applications
- Existing application COMMAREA, if used in the applications
- CICS BMS Map area(s)

This would result in a COMMAREA definition which would look like the following:

```
01 DFHCOMMAREA.
    02 STDHEAD.
    02 USR-TCTUA.
    02 USR-COMMAREA.
    02 INPUT-MAP-A.
    02 OUTPUT-MAP-A REDEFINES INPUT-MAP-A.
```

Layout of standard header

It is recommended that all applications use a standard header in their new COMMAREA definitions. The following example shows a recommended standard header:

```
* STANDARD CICS FIELDS
     02  NEW-HEADER.
         05  USRVERSION PIC 9(4).
         05  USRTRNID   PIC X(4).
         05  USRTRMID   PIC X(4).
         05  NXTRNID    PIC X(4).
         05  USRMAPNAME PIC X(8).
         05  USRMAPSET  PIC X(8).
         05  RETCODE    PIC 9(8).
         05  USRMAPLN   PIC 9(4).
         05  USRCPOSN   PIC 9(4).
         05  USRCALEN   PIC 9(5).
         05  USRAID     PIC X.
         05  ERASESW    PIC X.
         05  SENDTYPE   PIC X.
         05  ALARMSW    PIC X.
         05  FILLER     PIC X(23).
         05  TEXTAREA   PIC X(100).
```

All the fields are in COBOL character format. The header itself is 80 bytes long (including 23 unused bytes), and is followed by a 100 byte area to be used for EXEC CICS SEND TEXT commands. If longer text messages are required, then the TEXTAREA field should be expanded.

Many of the fields serve as replacements for the corresponding terminal-related fields in the CICS Execute Interface Block (EIB). The TEXTAREA field is used by EXEC CICS SEND TEXT commands. The USRMAPNAME field is used to indicate the BMS map that was specified in an EXEC CICS BMS command.

The USRVERSION field is optional. It can be used to indicate which version of a particular COMMAREA is being sent. With CICS 3270 applications, all programs and screen definitions are stored on the host. Therefore, it is fairly easy to coordinate changes in both the screen layout and the corresponding application programs. However, in a client/server environment, this is not the case. It can be very difficult to coordinate changes between all the clients which send in a particular COMMAREA, and the host program which receives and processes it. If the version number field is used, the host program can accept COMMAREAs in more than one format, allowing client changes to be phased in over time.

Rest of the COMMAREA

If the 3270 applications use a terminal user area (TCTUA), then this area should follow the standard header. Similarly, if the applications use a COMMAREA to maintain transaction context, then the existing COMMAREA should be included in the new COMMAREA.

Finally, any CICS BMS maps used by the application are added to the new COMMAREA definition.

Change the PROGRAM-ID paragraph

The converted program is generally given a new name, to avoid conflicts with the existing 3270 version of the program. Furthermore, if a new 3270 front end program is written (as recommended), then this program would be the logical one to use the existing program name. Therefore, the PROGRAM-ID paragraph at the beginning of the program is changed to reflect the new name of the program.

References to BMS Map Names

Typical CICS application programs include numerous references to BMS fields, to access user input and output fields, and to build the response to the user. BMS field names are of the format of a field name with various letters appended to the end, such as I, A or O. These references are difficult to change, since they are numerous and often intertwined with the application logic. For example, if a BMS field named ADDRESS is defined, then the various BMS variables for this field would be referred to as ADDRESSI, ADDRESSA, ADDRESSO, etc.

By using the same data structures and field names, references to BMS fields can be left unchanged. The client program is responsible for filling in any needed input fields, and can retrieve any output from the map area in the returned COMMAREA after the CICS transaction has executed.

It is recommended that the name of the BMS input map be included in the standard header, which should be checked in place of a BMS RECEIVE MAP command. The output map name would be returned to the calling program, to verify that the output is in the expected format. The map name field is needed to understand if a SEND TEXT command has been issued instead of the expected SEND MAP, and in cases where an application uses more than one map (the provided samples do use multiple maps in the same application).

Multiple maps in a single application

In some cases, a single CICS transaction may return different BMS maps, depending on application options. This can make use of the EPI difficult and error prone. By returning the map name in the COMMAREA, the client program knows which CICS map was used to map the data.

Terminal Control User Area (TCTUA)

CICS allows a small (up to 256 bytes) area of storage to be associated with a terminal. Since a user typically signs on to a terminal once, and uses the same terminal for all their transactions, this storage also stays with the same user. In the case of a non-terminal application, there is no equivalent storage area. The TCTUA definition should be placed in the new COMMAREA, probably after the standard header.

Any EXEC CICS ADDRESS commands which refer to the TCTUA should be removed from the application, and replaced with COBOL SET TO ADDRESS OF statements. Since a TCTUA definition would normally be an 01 level data structure in the LINKAGE SECTION of a COBOL program, it may be necessary to increase the levels of all individual field definitions by at least one. A utility is provided which does this, or alternatively, a global change function in a program editor can be used.

BMS Commands

A typical CICS program will issue a RECEIVE MAP statement near the beginning of the program, to receive any user input, and a SEND MAP statement near the end of the program, to send the results to the user's terminal.

EXEC CICS RECEIVE MAP

The EXEC CICS RECEIVE MAP command causes the input 3270 data stream to be transformed into a BMS data area. This statement is not needed, since the client application has provided the 3270 BMS input area directly in the COMMAREA, and should be removed. It is recommended that the RECEIVE statement be replaced with an IF statement to verify that the correct map name has been sent.

EXEC CICS SEND MAP

The EXEC CICS SEND MAP command causes a BMS data area to be converted to a 3270 data stream and sent to the 3270 terminal. This statement should be removed, since the mapped data stream is begin returned in the COMMAREA. It is recommended that it be replaced with a statement moving the map name into the header part of the new COMMAREA, and that an erase type switch be set if the ERASE or ERASEAUP options were specified, and a send type field be set if the MAPONLY or DATAONLY options were specified.

EXEC CICS SEND TEXT

The EXEC CICS SEND TEXT command sends a user data area to the 3270 screen. The data area may or may not include 3270 formatting characters. This command is often used to send messages to the user, especially error messages indicating unusual situations.

The map name field is used to indicate the use of EXEC CICS SEND TEXT. An additional send text message field is defined at the end of the COMMAREA, and a special map name would be used to indicate a text message is being returned rather than a BMS map.

To handle an EXEC CICS SEND TEXT command, the program would fill in a special value for the USRMAPNAME (SENDTEXT), and move the text to the TEXT-Message field (which must be long enough to handle the longest possible text message that will be sent using SEND TEXT). In the included sample, a 100 byte area is reserved for SEND TEXT messages. When the client program sees this special value in the Map-Name field, it knows to ignore the Output-Map-Area and instead get the message from the TEXT-Message field.

EXEC CICS RETURN with TRANSID and/or COMMAREA options

Pseudo-conversational programs often use EXEC CICS RETURN with the TRANSID, COMMAREA and/or LENGTH options. To handle these, the next transaction name should be passed back in the standard header in the COMMAREA. The current user COMMAREA can either be included in the new COMMAREA, or, if it is large, it can be saved in Temporary Storage and retrieved by the next transaction.

The USRCALEN field is changed to reflect the length of the new user COMMAREA. If an EXEC CICS RETURN is issued with no COMMAREA, then the USRCALEN field should be set to zero.

EIB Fields

There are several fields within the CICS Execute Interface Block (EIB) which application programs may use, but which either have no value or a different value in the case of a non-terminal transaction, such as an ECI application as opposed to a normal 3270 application. In particular, this would include:

- EIBTRMID - Terminal Name
- EIBTRNID - Transaction Name
- EIBCPOSN - Cursor Position
- EIBCALEN - COMMAREA length

- EIBAID - AID byte

Terminal Name (EIBTRMID)

A CICS program running under CICS/ESA, which is invoked by an ECI call, will not have a meaningful name in the EIBTRMID (terminal name) field. This field will usually contain an LU6.2 session number (usually a negative number, like -997) rather than a four character terminal name. Furthermore, the number is neither unique (many clients will see the same value) nor will subsequent transactions from the same client necessarily see the same value.

CICS 3270-based applications can access the terminal name. The terminal name is sometimes used to generate temporary storage queue names, log user access, or for other such purposes. In the case of a CICS client, a unique 4 character client, which can serve as the "terminal name", name must be added to the front of the COMMAREA.

There are several possible ways to assign unique names to the clients, which can be used in place of the CICS terminal name.

CICS clients allow for pre-defined names, by specifying a value in the CLIENT= parameter in the CICSCLI.INI file (located in the CICSCLI\BIN directory). This name can also be specified as an asterisk ("*"), in which case CICS will generate a unique name for the client. Unfortunately, there is no API on either the client or the server to extract this assigned name. If the client has a pre-defined name, the name can be extracted from the CICSCLI.INI file.

In some cases, client systems are identified by a unique name that is contained in the client application. In this case, this name should probably be used.

It is usually possible to extract either a host name or a TCP/IP address from the client. These can be converted into client names.

Lastly, a CICS program can be written, similar to an autoinstall user exit, which can be called using the ECI, and which assigns a unique four character name to each client which calls it. A very rudimentary sample of this technique is included in the VERIFYPW program shipped with this SUPPORTPAC.

Transaction Name (EIBTRNID)

When CICS programs are invoked from a CICS client using the ECI, they are usually run under a CICS transaction called the *Mirror Transaction* (CPMI). If the CICS application looks in the EIBTRNID field, it will see CPMI, and not the user transaction code that it is expecting to see. Therefore, any references to EIBTRNID are changed to references to a new field in the standard COMMAREA header, USRTRNID. If the CICS program uses this field, then the client application must fill it in with the appropriate user transaction code that the user application program expects to see.

It is possible to invoke the mirror program under a user specified transaction code. However, this transaction code would have to be different than the normal 3270 transaction code, since the original transaction code is presumably still needed for a 3270 version of the transaction. For further discussion of transaction codes, see the discussion in the advanced topics section.

Cursor Position (EIBCPOSN)

When a user sends 3270 input to CICS, the location of the cursor is placed in the EIBCPOSN field. For example, some menu programs allow selection by moving the cursor to a particular line or location and pressing enter. This can be handled by having the client program send in a location for the cursor.

A 3270 application can examine the location of the cursor when the user pressed the attention key to invoke the current transaction. Any references to the EIBCPOSN field will be changed to refer to the USRCPOSN field in the COMMAREA header, and the client application must then fill in the expected value.

COMMAREA Length (EIBCALEN)

Many pseudo-conversational CICS applications will leave a COMMAREA in CICS when they finish, which the next transaction will expect to find. Programs called via the ECI cannot leave a user COMMAREA in CICS between transactions, since the stored COMMAREA is associated with a terminal. Rather, the user COMMAREA must be passed from the client as input with the ECI request, and any user COMMAREA must be returned to the client application in the main COMMAREA when the transaction ends.

The existing user COMMAREA should be added to the new COMMAREA. Since the new COMMAREA must include other fields, the length of the COMMAREA, which is available in the EIBCALEN field, will include the other areas, such as the header and BMS map areas. If the application program is checking the EIBCALEN field, expecting it to reflect the length of the existing user COMMAREA, it will not find the expected value. For example, if it is checking for the existence of a user COMMAREA by checking for a zero value, then it will not see a zero value, even if there is no user COMMAREA. To ensure that the application continues to work as expected, a USRCALEN field is added to the standard COMMAREA header, and all references to EIBCALEN are changed to refer to USRCALEN. The USRCALEN field must be filled in by the client application, if the CICS application is expecting particular values in this field.

AID byte (EIBAID)

A 3270 AID (Attention Identifier) byte indicates which 3270 key was pressed by the user to send the input to be sent. Each 3270 key which causes input to be sent to the host has a unique AID byte value. For example, if the user presses the PF1 key instead of the ENTER key, a different value would be generated. If the CICS application can be invoked using different 3270 keys, the client program must enter the appropriate value in the USRAID field in the COMMAREA.

The EIBAID byte normally contains the value for the 3270 key which the user pressed to invoke the current CICS transaction. Since there is no 3270 terminal, this field will not contain the expected value. Thus, any references to this field are converted to USRAID references, and the client application must fill in the appropriate value into this field. The client application should be aware of the data translation normally performed on CICS COMMAREA fields, and should thus use the ASCII equivalent of the AID value.

Symbolic values for the AID byte values are provided in the CICS_EPI.H file, found in the \CICSCLI\INCLUDE directory.

EXEC CICS HANDLE CONDITION MAPFAIL and HANDLE AID

Both of these EXEC CICS HANDLE statements cause applications to go to specific routines after the execution of an EXEC CICS RECEIVE statement. To process them, two fields are added to the standard COMMAREA header, namely a map length field, for the MAPFAIL condition, and an AID byte, for the HANDLE AID statement.

Within the CICS API, an EXEC CICS RECEIVE MAP will raise the MAPFAIL condition if no input data is received, which includes the case of the user pressing the clear key. This condition would occur if a user presses the ENTER key without typing any data into the screen. A typical application use might arise in certain data editing situations, where the application keeps checking the data and displaying it, until the user presses the ENTER key without any changes to the fields on the screen. By this action, it is assumed that the user does not want to make any further changes, and processing of the data on the screen can proceed.

To properly simulate the MAPFAIL situation, an IF statement must be inserted in place of the EXEC CICS RECEIVE statement (which must be removed to convert the program), to check if the map length field is zero, and if it is zero, to go to the label specified on the MAPFAIL command.

In a similar vein, the HANDLE AID command is handled by inserting additional IF statements, one for each key specified in the HANDLE AID statement, which go to the respective locations depending on the value in the aid byte field.

The supplied conversion utility (CBLCVRT) makes a simplifying assumption. If it finds a HANDLE CONDITION MAPFAIL or HANDLE AID statement, then any statements in the program which follow the HANDLE statement will have the appropriate IF statements added. This handles the most common uses of these statements. However, the program logic of the application should be reviewed to see if the statements have been inserted appropriately. If the RECEIVE or HANDLE statements are in a separate program, or if they are not in sequence, then the necessary IF statements will not be inserted automatically by the conversion program, and they must be inserted manually.

Furthermore, the HANDLE statements can be conditionally executed, depending on conditions at run time. In these cases, a more sophisticated approach is required, using additional variables to determine if the HANDLE statement is in force when the RECEIVE statement would have been executed. It is believed that such usage is rare, and must be handled manually.

Manual changes

For various technical reasons, not all required changes are made by the conversion utility (CBLCVRT). The intention is to automate as much of the conversion as is practical without causing too much complexity and therefore making the conversion utility itself hard to use or understand.

BMS copy statements must be removed from the WORKING-STORAGE section, and re-inserted following the copy statement for the standard header. The names of the BMS copy files should probably be changed, since the levels of the BMS fields must be changed. This avoids conflicts with the existing BMS copy files. An alternative to this approach is discussed in the section about writing a new 3270 front end.

If a user COMMAREA is present and the definition of the fields is contained in a copy book, then the levels of the individual entries in the copy book may have to be increased, as with BMS field definitions. If no copy book is used, but the fields are defined directly in the program, then the level numbers should be examined and adjusted manually if necessary.

Summary

At this point, most or all of the 3270 BMS dependencies in the existing program have been handled, without requiring major changes to the CICS application, and without introducing major complexities on the client side.

Sample Programs

As part of this SupportPac, two of the Installation Verification Programs (IVP) provided with the IBM Transaction Server of Windows NT™ have been selected as samples, and converted. This chapter will cover the conversion process of the selected samples in detail.

Overview of sample programs

The IVP programs shipped with the IBM Transaction Server for Windows NT™ consists of five CICS COBOL programs, plus five associated BMS maps. The programs are named as follows:

- DFHCMNU.CCP - Menu program
- DFHCALL.CCP - Main Add/Update/Inquire program
- DFHCBRW.CCP - Browse program
- DFHCREN.CCP - Simple part order program
- DFHCCOM.CCP - Order entry queue Print Program

The maps are named as follows:

- DFHCGA.BMS - Menu map
- DFHCGB.BMS - Detail display map
- DFHCGC.BMS - Browse display map
- DFHCGK.BMS - Order Entry map
- DFHCGL.BMS - Order Entry printer map

For this example, only the DFHCALL.CCP and DFHCBRW.CCP programs will be converted.

The menu program is very small and does nothing more than display a menu map. It is replaced by function in a new GUI or web front end.

The DFHCALL and DFHCBRW programs are reasonable samples of CICS COBOL programs. They manipulate a single indexed file (FILEA), offering inquiry, update, add and browse functions. The fields in the records consists of an account number, name, address, phone number, date, amount field and comment field. The individual records are 80 bytes long. As installed, the data file does not contain any records.

The DFHCBRW.CCP program that is supplied with the IBM Transaction Server to Windows NT is a conversational program. To make this program suitable for the technique described in this SupportPac, the program was first converted to be pseudo-conversational. The converted program (DFHCBRW.CCP) is supplied with this SupportPac, and should be used in place of the version supplied with the Transaction Server. While the conversion was done in this case, conversion of conversational programs to be pseudo-conversational can be difficult, and it is generally recommended that the technique described in this SupportPac should not be used with conversational programs.

It is suggested that the IVP program be installed and run as a 3270 transaction, and that some data records be added. This will create the data file and compile and install the associated programs, maps and CICS table entries. To execute the installed transactions, start a terminal emulation session, and type the word MENU (in capitals) and press enter (usually the lower right control key in a 3270 terminal session). To add some records, enter the transaction code ADDS in the first field on the menu screen, and an account number of up to six numbers in the account field. Fill in the fields on the detail screen and press enter to add the record. It is recommended that at least five or six records be added in this manner, so that there is some meaningful data to access. Please be aware that the phone field should be of the format nnn-nnnn, where n is a number, the date field should be of format nn/nn/nn, and the amount field should be of format $nnnn.nn (note the dollar sign and decimal point), or the data editing routines will reject the data (in a rather unfriendly manner).

It is recommended that the version of the DFHCBRW.CCP program be used instead of the normal one supplied with the Transaction Server Installation Verification Programs (IVP), since this program has already been converted to operate in a pseudo-conversational manner rather than conversationally.

DFHCMNU program overview

The dfhcnmu.ccp program is a menu program. It displays a menu using the MENU map in the DFHCGA mapset and then exits. It is invoked via a MENU transaction code. The user then selects one of four transaction codes, namely INQY for inquiries, BRWS for file browse capability, ADDS for file adds and UPDT for file updates. The INQY, UPDT and ADDS transaction codes invoke the DFHCALL program, while the BRWS transaction code invokes the DFHCBRW program.

The menu transaction is not needed in a client environment and thus will not be changed.

DFHCALL program overview

The dfhcall.ccp program handles inquiry, add and update requests. It expects to be called under one of three transaction codes, namely INQY for inquiries, ADDS for add requests or UPDT for update requests. It looks for a previous commarea left from previous invocations. This transaction is pseudo-conversational.

DFHCBRW program overview

The dfhcbrw.ccp program will initiate a CICS browse operation against the FILEA file. It can browse forward or backward, depending on the function key that was pressed (PF1 to page forward or PF2 to page backward). The clear key is used to cancel the current browse operation. Up to 4 records at a time are displayed. The program as supplied with the Transaction Server is conversational. Please use the version of this program supplied with this SupportPac, which has been converted to be pseudo-conversational. It uses a map called BROWSE in the mapset DFHCGC to return the data. Once the browse operation is started, then the direction is controlled by the input field DIRI ('F' for forward browses, 'B' for backward browses).

DFHCREN program overview

The dfhcren.ccp program accepts order entry data (part number and quantity) and stores it in a temporary storage queue named L860. The TS queue records are defined in a copy book named dfhcl86. It uses a BMS map called ORDER in mapset DFHCGK. The order can be cancelled by pressing the clear key. If the order is accepted, then the screen is cleared and a text message ("PROCESSING COMPLETED") is sent to the screen. This program is conversational. Simple editing is performed and if the data does not pass the edit, then the field with the error is highlighted and the user is asked to correct it.

DFHCCOM program overview

The dfhccom.ccp program prints the order that is saved in the TS queue L860. This program expects to be run from the printer L860. If it is run from a different terminal, then it schedule an OREQ transaction to print the order after an interval of 10000 seconds. If the terminal is in fact L860, then it will print all of the orders in the queue.

Conversion Procedure

Make copies of the programs and BMS maps to be converted

At this point, make copies of the two CICS COBOL programs, as well as the the BMS map copy files and the other required copy files. It is recommended that a new directory be created, and the files copied into that directory. For example, a directory of name TESTCBL could be created from a command prompt by the following command:

```
MD \TESTCBL
CD \TESTCBL
```

The files could then be copied in as follows, assuming that the transaction server product has been installed on the c: drive.

```
Programs
  copy \opt\cics\src\samples\ivp\dfhcall.ccp \testcbl
  copy \opt\cics\src\samples\ivp\dfhcbrw.ccp \testcbl
BMS Maps
  copy \opt\cics\src\samples\ivp\dfhcga.bms  \testcbl
  copy \opt\cics\src\samples\ivp\dfhcgb.bms  \testcbl
  copy \opt\cics\src\samples\ivp\dfhcgc.bms  \testcbl
COBOL copy books
  copy \opt\cics\src\samples\ivp\dfhcfil     \testcbl
  copy \opt\cics\src\samples\ivp\dfhclog     \testcbl
```

It is recommended to use the modified pseudo-conversational version of the DFHCBRW.CCP program rather than the one supplied with the Transaction Server product.

Process the BMS map source to create COBOL copy books

At this point, the bms maps should be processed to create the necessary COBOL copy books, as follows:

```
cd \testcbl
cicsmap dfhcga
cicsmap dfhcgb
cicsmap dfhcgc
```

This step uses the CICSMAP function provided by the IBM Transaction Server for Windows NT™. This creates copy books for the BMS map areas. As an alternative, if the copy books already exist, then they can be copied into the TESTCBL directory

Map conversion

In order for the data structures created by the BMS map processing to be used in a CICS COMMAREA, the COBOL levels of the structures must be increased by at least one, so that the level of the BMS maps starts with at least an 02 level rather than an 01 level. A utility (CHGMAP) is provided to increase the levels of all variables in a COBOL copy book by one. The utility takes as input the name of a COBOL copy book file, and generates a new copy book file with a file extension of NEW. This file should be renamed as appropriate. To raise the level of the field definitions in the BMS map copy books, using the CHGMAP utility, the following statements could be used:

```
cd\testcbl
chgmap dfhcga efhcga
chgmap dfhcgb efhcgb
chgmap dfhcgc efhcgc
```

Run the Migration utility

The next step is to use the CBLCVRT utility to make most of the necessary changes to the COBOL programs. The utility will product a new text source file with a file extension of NEW. This file should then be given a new name, and the extension changed to CCP. This could be done as follows:

```
cd \testcbl
cblcvrt dhfcall.ccp newcall.ccp efhcall.ccp
cblcvrt dfhcbrw.ccp newcbrw.ccp efhcbrw.ccp
```

Define a new COMMAREA

The conversion program will insert a COPY statement for a copy book named STDHEAD. This is intended to contain the standard header and potentially the TCTUA. Any copy statements for the DFHCOMMAREA definition will be left in place. It is necessary to insert copy statements for any BMS maps that are used by this program or any programs which are called by this program. In addition, the BMS map copy books themselves must have their level numbers increased by at least one.

If no LINKAGE SECTION or DFHCOMMAREA statements are found, then they will be inserted by the program automatically.

Review the changes that the conversion utility made

It is recommended that the message file (MSG file extension) produced by the conversion program be examined, to see what changes were made and what problems (if any) were detected. The file also contains some statistics and any error messages that were generated. For example, to view the message files using the Windows notepad facility, type the following:

```
notepad dfhcall.msg
```

Make any manual changes that are necessary

The conversion utility does not make all the necessary changes. Therefore, please start up a program editor (the Notepad or Edit programs, which ship as a standard part of Windows NT, can be used), and make the following changes.

1. Move the copy statements for the CICS BMS areas from the WORKING-STORAGE SECTION to follow the DFHCOMMAREA statement.

In the case of the DFHCALL program, there are two BMS map copy statements that must be moved from WORKING-STORAGE to the LINKAGE SECTION, namely DFHCGA and DFHCGB.

The DFHCBRW.CPP program is more complicated, since this program as written is a conversational program. This program would probably not be acceptable in a production environment, since the program has a browse operation active across user interactions, and the browse operation holds exclusive control on part of the user file. Thus, a user could cause other user's terminals to lock-up if they started a browse and then left it active for an extended time.

Conversational programs are not normally good candidates for this process, since they must be converted to be pseudo-conversational, and this can be quite difficult in many cases. However, in this case, primarily because the program is a sample program, the migration is not particularly difficult, and has been done. This was done by hand.

Documenting the new callable interface

Now that a new interface has been added to the program, it is a good idea to document how to use it. In particular, the COMMAREA input and output should be described, and any required input fields noted.

In the case of the DFHCALL sample program, three functions are provided, namely the ability to inquire, update and add a record into a file. By examining the message file produced by the conversion program, the program uses the EIBTRNID field and the EIBCALEN field. The EIBTRNID field is used to determine the function to be performed (for example, inquire, update or add), and the EIBCALEN field is used for update and add requests to determine if this is the first or second invocation of the transaction (a user COMMAREA is left by the first invocation, and used by the second invocation to ensure that the record has not been changed by another user between invocations). Therefore, the USRTRNID and USRCALEN fields must be filled in, and are part of the interface definition.

The program uses two different maps. The menu map is normally used by the first pass. It contains only a transaction code (EIBTRNID) and account number. The second map is a detail map, providing the information in the account record. The menu map is also used to send messages back to the user (such as "RECORD UPDATED").

Therefore, the interface definition would normally look something like the following:

> For inquires, the following fields are required and must be filled in using normal COBOL conventions.
>
> Output from the conversion utility can be very useful in documenting the CICS fields which a client application must fill in to drive the converted application. For example, if references to the user transaction code (EIBTRNID) were found and converted to references to USRTRNID, then the client program must insert the correct transaction code into the USRTRNID field, and this should be included in the documentation.

Writing a new 3270 front end

After a program has been converted, and given a new name, there are now two programs with the largely the same business logic. This can increase maintenance costs, since two programs must now be maintained. It is usually possible to write a fairly simple 3270 front end program which handles the 3270 user input and output, using BMS, and does an EXEC CICS LINK to the new program, passing an appropriate COMMAREA. This program usually does an EXEC CICS RECEIVE MAP into the appropriate input area, builds the rest of the COMMAREA from the EIB, TCTUA, etc, and then calls the converted program. A working sample if provided as EFHCALL.CCP, which is a new 3270-based front end program to the converted DFHCALL.CCP program, NEWCALL.CCP.

The conversion program can optionally generate a skeleton 3270 front end program, if the third parameter is specified. The generated program is based on a template file called NEW3270.TXT. This file contains a COBOL program. The conversion program will make changes to four areas of the template to produce the skeleton 3270 front end program, namely:

1. Change the PROGRAM-ID to match the name of the new 3270 BMS front end. The new name is taken from the third parameter, and is the file name with any leading directory and trailing file extension information removed.
2. Change the name of the linked to program to match the new name of the converted program.
3. Replace the comment about inserting any BMS RECEIVE statements with any receive statements found in the program.
4. Replace the comment about inserting any BMS SEND statements with any SEND statements found in the program. The SEND statements will be surrounded by IF statements to insure that the correct SEND statement is used, based on the characteristics of a particular send.

There is some manual fix up required to produce a working program. Again, any BMS copy statements must be added to the new COMMAREA to be passed to the converted program. In addition, if more than one BMS RECEIVE statement is found, conditional logic must be added to the program to ensure that the correct RECEIVE statement is used. A warning message is issued to this effect if more than one RECEIVE statement is found. The reason for this is that a CICS program can contain more than one RECEIVE statement, with conditional logic to only execute one of them. For example, the sample programs contain logic to do a BMS RECEIVE for either the MENU map or the DETAIL map, depending on the size of the user COMMAREA. This logic must be added to the generated skeleton program for the program to work correctly.

Other manual changes may be required to handle unique requirements or program logic.

ECI Password Verification

A CICS COBOL program (VERIFYPW) to perform user id and password verification has been provided. This program performs several useful functions, including:

- Verification of a passed user id and password
- Assigning a unique 4 character "client name", if needed
- Logging of the user id and client name

The verification of the user id and password uses the CICS/ESA V4.1 interface EXEC CICS VERIFY PASSWORD. This program is intended to operate in a CICS/ESA V4.1 or higher environment, or in other CICS environments which support the 4.1 level of API, such as CICS for OS/2 V3. This API is not available in the Transaction Server for Windows NT V4™ product or the related Unix-based products. To execute the

VERIFYPW program in this environment, it is necessary to either remove the EXEC CICS VERIFY PASSWORD statement, or to configure the system so that this program is located on a CICS/ESA V4.1 (or higher) system, and configure transaction server to forward requests for the VERIFYPW program to the corresponding host. It is also a good idea to define a special user id with very limited authority to execute the VERIFYPW program under.

VERIFYPW does not actually cause a "sign-on" of the CICS client. Rather, it checks if the user id and password provided are in fact valid. The VERIFYPW program itself should be configured to run either with no user id and password, or with a fixed user id and password which can only execute the VERIFYPW program itself. This is similar to the situation with the CICS sign on program, CESN, which must be able to run without the user being previously signed on.

The VERIFYPW program also performs logging of the verification event. It writes the user id and client name to the CICS console log, using the EXEC CICS WRITE OPERATOR API statement. Again, this statement is not supported by the NT Transaction Server product. If this program is to run on NT Transaction Server, then this statement should be removed.

The last function provided by this program is to assign a unique client name to the client. Please note that this function uses the first four bytes of the CICS Common Work Area (CWA) for this function. This therefore requires that the CWA in fact be defined as at least four bytes long on whatever system the VERIFYPW program runs.

If no CWA is defined, then the VERIFYPW program may abend. Be aware of any conflicts with other applications using the CWA. If other applications need to use the CWA, then the location of the 4 byte field must either be changed, or this function removed. This function is only invoked if no client name is passed in with the logon request. In some environments, a client name may be defined locally on the client system, and this name passed in with the verification request. In this case, no name is generated, nor is the CWA area used.

This program is also a good place to perform other application "initialization" types of processing, which might be triggered by the autoinstall of a client or some form of sign on in the 3270 environment.

This program should be unprotected, since it functions similarly to a sign on program (such as CESN).

Advanced Topics

Application Testing

Use of Temporary Storage and CECI

One area of difficulty with non-terminal applications is testing. They usually require an actual client program to invoke them properly, and this usually makes it difficult to stop the application with a debugger when it is invoked.

CICS provides a CECI transaction, which can invoke a back end CICS program by using a LINK statement, and passing a user defined area as storage. However, the layout of some COMMAREAs can be rather complex, and therefore entering the COMMAREA into the CECI transaction can be difficult. Fortunately, there is a fairly simple technique to get around this.

To get around this problem, write a very simple CICS application which merely captures the COMMAREA and writes it to a known temporary storage queue. This program should have a unique name. It does depend on the COMMAREA, so only a single version is required for testing any back end transaction. Modify the client program to call this test program rather than the real CICS back end (this usually involves changing a constant). This allows the COMMAREA, loaded with real data, to be captured. After the COMMAREA is captured in temporary storage, start a CICS terminal (CICSTERM with the CICS client) and execute the CECI transaction. If use of an online debugger is desired, such as the CICS supplied CEDF facility, it should be started before the CECI transaction. Next, do a temporary storage read into a named user variable. Finally, invoke the real CICS back end transaction using CECI and a CICS LINK, pointing to the user variable with the COMMAREA image that was loaded from temporary storage. A sample COBOL program to capture the COMMAREA into temporary storage is provided is provided. The program is named CAPTURE.CCP.

Menu Programs

Many menu programs can be adapted without undue effort. There are a couple of things to look out for. Some menu programs use either EXEC CICS RETURN IMMEDIATE, possibly with the INPUTMSG option. They may also do a EXEC CICS START NEXT. These techniques are in conflict with the CICS ECI, and cannot be used.

Regardless of whether these programs use techniques which do not work with the ECI, it is recommended the menu functions be moved to the front end GUI or Web interface, rather than continue to be driven by CICS. For cases where a dynamic menu is presented to the user, some interaction with the CICS host may still be required. However, this can usually be accomplished with a fairly simple host program, which returns the appropriate menu options.

Many CICS users start with an initial sign on type of screen, and are then taken directly to a menu screen. Both of these functions can be handled by a front end interface, accompanied by relatively simple back end CICS programs. For validation of a user id and password on a back end host, a simple CICS transaction can be written which uses the EXEC CICS VERIFY PASSWORD function (introduced in CICS/ESA V4.1) to check the validity of a user id and password. The client program can then proceed directly to menu options.

COMMAREA programming considerations

Data formats

If COBOL is used at both ends, then the data formats are basically the same and the programmer does not need to do anything special. However, if a different language is used on the client side, as is usually done, there are some differences in the fundamental data types that must be understood. The differences can be handled with common subroutines, which minimizes the programming burden. Working versions of these subroutines are provided with the companion client SupportPacs.

CICS programs generally deal with three primary data types, strings, integers and packed decimal fields. Strings can be further broken down into two main types, numeric and alphameric strings. The main difference between numeric and alphameric strings is justification and fill character.

On the PC side, there are two general data types that are dealt with, strings and integers. The strings are different from COBOL strings, using zero (null) delimiters rather than being a fixed length with padding.

Integer formats are generally quite compatible, with both languages supporting 16-bit (short) or 32-bit (long) formats. Any differences in byte ordering ("big-Endian/little-Endian") can be handled with the CICS data translation mechanisms on the host (see the DFHCNV Conversion Table Generation utility).

Most languages other than COBOL (including C, Java and Visual Basic) do not have native support for packed decimal formats. This requires common subroutines to convert between either numeric strings or integer formats on the PC side and packed decimal on the host side. This should not be an issue for most COMMAREAs covered by this SupportPac, since the standard header and BMS map areas do not use packed decimal, and the user COMMAREA in most cases is not used by client programs.

BMS fields

In addition to providing the actual data that a user has entered into a terminal, CICS Basic Mapping Services (BMS) also provides additional fields. In particular, BMS provides a length field which indicates the number of characters a user typed into a particular field. If a user did not enter any data into a field, then the length field is zero. Furthermore, if the user enters no data into a field (or presses the clear key), then any BMS RECEIVE command will raise a MAPFAIL condition.

To simplify the process of filling in BMS map fields, additional common subroutines have been provided, which in addition to filling in the data portion of the map field (which is always a COBOL string), also automatically fill in the BMS map field length and add the length to the total data length for the map.

User COMMAREA

Many CICS programs are expected to be run in a particular sequence, and save data between transaction invocations within CICS itself (sometimes referred to as context). This data is really associated with a CICS terminal. This technique will not work in a CICS ECI environment, for two reasons. First, there is no terminal to associate the data with, and second, the user program is not the highest level CICS program (the IBM supplied mirror program is).

The other difference is that a program called via the ECI always has a COMMAREA. Thus, if an application is looking at the EIBCALEN, particularly for a zero value, it will not see the correct value, since other areas have been added to the COMMAREA. For this reason, a USRCALEN variable has been added to the standard header, and this variable will contain the length of the user COMMAREA, rather than the length of the entire COMMAREA.

Maximum COMMAREA size

CICS COMMAREAs passed using the ECI are limited to 32,500 bytes of user data. If more data needs to be passed, then the data must be segmented by the application, and multiple ECI calls made, probably using extended calls.

In most cases, the 32,500 byte limit is not likely to cause problems, and particularly for converted applications. BMS map areas are typically 1-2,000 bytes long, so that a typical COMMAREA length for a converted program is likely to be in the 2-5,000 byte range.

Other factors to consider would include the size of a screen, and the speed of the communications facility. Screens today typically display about 2000 characters of data, and this has not changed appreciably over time. Given white space and label requirements, it is unlikely that much more than 1,000- 1,500 characters of user data can be displayed on a single screen. That means that a 32,000 byte COMMAREA can typically hold 20 to 30 screens of user information. Users rarely want this much information from a single interaction.

As for line speeds, a typical 28.8 KBPS modem transfers data at about 2,000 bytes per second. That means that a single 32,000 byte COMMAREA would take over 15 seconds of dedicated line time. If all users are connected with LANs or high speed lines (for example, T1 lines), it is certainly possible to move more data in reasonably short times. However, applications much handle users at home, on the road, etc, where high speed communications may not be realistic.

Extended Calls

CICS clients provide the ability to issue multiple ECI calls, and treat them as a single transaction, using a single unit of work. This facility is known as extended calls. In the case of extended calls, the mirror transaction is attached with the first call, and remains in place for the duration of the transaction sequence. To end an extended sequence, a call must be made without setting the extended call attribute, or a COMMIT call must be made. A COMMIT call is basically an ECI call where no user program is executed.

Since the mirror transaction remains in effect, the security information and transaction name must be specified with the first call, and are ignored on subsequent calls. There is also some reduction in overhead, since the transaction is already running and need not be started, and the security context is already established, etc.

Extended calls allow for multiple CICS transactions to be combined into a single unit of work. What used to be separate units of work can now all function as a single composite unit of work.

One note of caution should be mentioned. Extended calls should not be used over user interactions or other times when they might result in a significant delay. Since the mirror transaction does not end after the initial call(s), locks on CICS resources may be held by the transaction, and if the transaction is not completed in a short time, other users could have their CICS transactions appear to lock up. This is the same danger as exists with conversational transactions.

Data translation

User data in a 3270 data stream consists of a limited set of character data. The data can be translated from EBCDIC to ASCII automatically. However, since the ECI passes user defined data areas between a workstation (generally ASCII) and a host (EBCDIC), and the data area is not limited to character data, data translation cannot be handled automatically. CICS provides a mechanism to define the layout of the user data area (COMMAREA) on the host, which allows for data translation to be provided. This uses the DFHCVT mechanism on the host. A template definition is required for each different host program which may be called by a workstation client using the ECI.

Security

The ECI provides for a CICS user id and password, which are checked on each ECI call (or the first call for an extended unit of work). This differs from classical CICS signon security, where a sign on transaction is executed once, and the terminal is then signed on.

There is a legitimate concern about the overhead of doing a sign on for each transaction. While path length numbers are never precise and can vary for many reasons, a typical path length to verify a user id and password, using IBM RACF, is about 90K instructions. Given that a typical CICS transaction takes about 700K instructions, this would appear to imply an overhead of about 13%. However, the newer releases of CICS/ESA cache recently used user ids and passwords, reducing the overhead to approximately 10K instructions (between 1 and 2 percent, which generally cannot be measured).

CICS/ESA V4.1 introduced EXEC CICS VERIFY PASSWORD. This API will take a user id and password, and verify if the pair is valid. It can also optionally return some other security related information, such as the number of days until the password expires, number if invalid logon attempts, etc. This can easily be incorporated into a simple ECI callable program on the host, which can be used by a front end GUI to accept and then validate immediately a user id and password. The user id and password can then be retained on the client and sent with each ECI request, effectively "signing on" the client.

There is one other aspect of security that should be considered. Many CICS clients are actually attached to an intermediate CICS system other than the host, with all requests forwarded to and processed on the host. For example, the intermediate CICS system could be CICS for OS/2, CICS for AIX, CICS for Windows NT, etc. The ECI request will actually arrive at the intermediate CICS, and then be forwarded on to the host system. The most common reasons for using an intermediate CICS system are for TCP/IP (or possibly NetBIOS) client connections, session concentration and to eliminate the need to define all clients on the host system. For CICS systems prior to CICS/ESA V4.1, LU 6.2 connections with parallel sessions cannot be autoinstalled (to directly connect a CICS client to a host system requires an LU 6.2 connection, preferably with parallel session capability).

The security problem comes in when the intermediate CICS system tried to check the user id and password. While all the intermediate CICS systems have the capability to validate user ids and passwords, this means that the user ids and passwords must be defined and maintained in two locations, using two different mechanisms (for example, on the host, using RACF or the equivalent, and on the intermediate CICS system). Most commonly, this means that all client user ids and passwords must be defined in the SNT (CICS for OS/2) or User Definition (UD) stanza file (CICS for AIX and CICS for Windows NT) on the intermediate system. The capabilities of these three intermediate systems are significantly different, so each will be covered.

CICS for OS/2 can store user ids and passwords in either its own Sign-on table (SNT) or in the OS/2 based UPM. It also provides for user exits which can decide whether or not to accept a particular user id and/or password. Finally, it provides a program which can use the Password Expiry Manager (PEM) facility of CICS/ESA introduced in CICS/ESA V3.3. This allows CICS for OS/2 to send user ids and passwords to the host system for validation. The sign on user exit allows a CICS for OS/2 system to be configured to accept any user id and password for a non-facility transaction (such as an ECI call). The user id and password will then be forwarded to the host when the request is forwarded, and the host will verify the user id and password. If local processing is also required, then the PEM mechanism can be used to verify the user id and password on the host. This allows user ids and passwords to be defined only once, on the host.

CICS for Windows NT V4.0™ provides the ability to remember but not check user ids and passwords on ECI requests. If the request is then forwarded on to a host system, the user id and password will then be passed on to the host for verification. Again, this allows user ids and passwords to be defined only once, on the host. This is what is required for gateway configurations. It is a problem only if local processing with security is required as well as forwarding of requests to a host.

CICS for AIX only supports identify (only user ids which have been verified locally) security for host connections. This means that for an ECI request, the user id and password will be checked by CICS for AIX, and must be defined in the CICS for AIX User Definition (UD) file (or in both the UD file and in DCE, if DCE security is used). This means that user ids and passwords must be defined on both the host as well as on the CICS for AIX system, even if the CICS for AIX system is merely functioning as a gateway or session concentrator.

Also, be aware that with CICS for AIX, the CCIN transaction is protected by default. The CCIN transaction is executed when a CICS client connects to a server (CCIN can be thought of as a replacement for negotiable bind for non-SNA client connections). This will generally cause a pop-up window to appear (which sometimes fails to seize the focus, and winds up hidden under other windows) when the first ECI call is attempted, requiring the user to enter a valid CICS for AIX user id and password. This can be circumvented by either changing the definition of the CCIN transaction or ensuring that a valid CICS for AIX user id and password are provided, either on the initial ECI call (if the client connection is started implicitly) or on the CICSCLI command if the client connection is started explicitly.

The following technique can help to minimize the problem of user ids and passwords having to be defined to both the host and CICS for AIX. First, code a user transaction on the host, which accepts a user id and password in a COMMAREA, executes a CICS VERIFY PASSWORD command, and passes the results back to the caller. Like the CICS sign on transaction, this transaction must be unprotected. A separate user file or table is maintained on the host, with a CICS for AIX password for the user id that is to be used on subsequent protected ECI calls. This password (and possibly user id) are passed back to the front end program, and are used on all subsequent ECI calls (which are protected transactions). This allows the CICS for AIX password (and possibly user id) to be hidden from the user, and largely maintained by some kind of automated procedure.

Additional discussion of security is covered in the section on "private mirrors" below.

User Transaction Codes (Private Mirrors)

An ECI call, regardless of which CICS program is to be executed, will request the default CICS internal transaction code of CPMI (function ship transaction from an ASCII client). CPMI must point to the appropriate CICS mirror program (name varies by platform). This can result in problems in three areas, namely:

1. Security - since most installations use transaction and not resource security.
2. Accounting - all ECI transactions run as CPMI. Transaction accounting is lost.
3. DB2 plan binding - all DB2 plans must be bound to CPMI.

To solve these problems, CICS supports the concept of so-called private mirror transactions. An installation can define their own transaction code, which must point to the CICS mirror program as the program to execute (in the PCT). When support for private mirror transactions was first added, it solved the above problems. However, it created a new problem with data translation.

When ASCII based versions of CICS were first created (CICS for OS/2 V1), function shipping transactions required data translation when requests were sent from ASCII systems to EBCDIC systems (and vice versa). This was solved by using a different transaction name (CPMI vs CVMI) for the function ship request, and by modifying the CICS mirror program to look at the transaction code which was used to invoke the mirror program. If the transaction code was CPMI, then translation was performed. If not, then no translation was performed. As a result, if private mirrors were used from ASCII to EBCDIC systems, the transaction code was not CPMI. Data translation was not performed by CICS, and had to be added to the application. To fix this problem, CICS has added PIP data to the function ship request.

PIP data consists of extra data which flows in the FMH5 attach header. The PIP data identifies the code page characteristics of the calling system. The receiving system can then examine the PIP data as well as the transaction code, and decide if data translation is in fact needed. PTFs were provided for all then current CICS host systems, including CICS for MVS V2.1.1. In addition, support for sending of PIP data was added to the CICS Common clients (in CSD 1), CICS for AIX V2.1, and CICS for Windows NT V4.0™. It should be noted that CICS for OS/2 V3.0, as well as some SNA products, do not support the sending of PIP data. In particular, the Microsoft SNA Server product does not support sending of PIP data. IBM Communication Server for Windows NT™ does support PIP data. The main impact of the lack of support in the Windows SNA products is when Windows clients are directly attached to a host system (using SNA).

A more common method of connecting CICS clients is to connect to an intermediate server, on either an OS/2, AIX (or other Unix - not discussed further) or Windows NT platform. The clients use TCP/IP to connect to the intermediate CICS server, and ECI requests for the host are routed on to the host. All the intermediate server products provide a mechanism to define CICS programs as remote, and to provide an alternate user-defined transaction code to be used instead of the default of CPMI. These alternate transaction codes must be defined on the on the host as private mirror transactions. CICS for AIX and CICS for Windows NT V4.0™ when using IBM Communications Server will automatically append the requisite PIP data when forwarding requests on to a host system.

Known Conversion Problems

The conversion techniques described in this SupportPac should be useful for a substantial majority of existing CICS applications, namely 3270 COBOL applications using BMS. The techniques themselves, but not the supporting utilities, should be useful with applications written in other languages. However, there are certain programming techniques which do not fit the model. If these techniques are used, then use of the techniques described in this SupportPac is not recommended.

What sorts of things cause major problems?

1. Conversational programs (see discussion of Call-Return Model)
2. Printing and Unsolicited Messages
3. Use of certain BMS options
4. Use of "terminal front end or menu programs" rather than BMS
5. Lack of source code - for example, a purchased package or languages other than COBOL

In cases such as these, the use of the EPI may be necessary.

Conversational Programs

The flow of 3270-based CICS applications is controlled by the host, usually using the EXEC CICS RETURN command with the TRANSID option. However, client/server applications are generally driven by the client. This paradigm change requires close coordination between the host and client sides. In particular, it is desirable to pass back the next program that the host expects to the client.

The ECI is a so-called CALL-RETURN model. That is, a program is called from the client and must complete its execution before returning data to the client. The called program cannot contact the client in the middle of its execution. A client can issue multiple calls within the same transaction (Extended Calls), but each program must complete before the next program can be called. Pseudo-conversational transactions generally have this characteristic. Conversational transactions do not. Fortunately, for various technical reasons, most CICS programs are written as pseudo-conversational transactions.

There is no simple solution for conversational programs. The programs must be converted to be pseudo-conversational, and this step may well prove very difficult. The EXEC CICS SEND MAP/RECEIVE MAP (or CONVERSE) must be commented out (easy part) and the program must be made pseudo-conversational (not easy). In some cases, this can be fairly easy, such as cases where the conversational request is to confirm some action. The confirmation can be moved into the front end CGI program (or client program), where it logically belongs, and the program logic left intact.

Printing and Unsolicited Messages

There is an inherent conflict between CICS applications, where the server controls the application flow, and client programs, where the client controls the application flow. In particular, printed output which originates at the host can be a problem, since the host cannot originate an exchange with the client. This generally requires some re-architecting of the application.

In the case of Web based applications, the basic architecture does not support the notion of a server originating an exchange with the client. Rather, the client must originate all exchanges. There are some more recent developments, generally called "push" technologies, which do provide potential solutions for some applications in this area.

In the case of so-called "Fat Client" applications, there is a potential solution. Both CICS clients and servers support the notion of printers. In the case of CICS clients, printers can be defined, and can receive unsolicited output from CICS server systems. There is no requirement to limit use of this facility to printed output. The CICS client printer support accepts the output of a CICS printer, and then invokes a user specified program to process it. While the invoked program might send the data to a real printer attached to the client, this is not necessary, and in fact the program could do just about anything on the client. Thus, this facility can handle not only printed output, but any server initiated functions to be performed on the client.

Printing is another area of potential problem. Browsers provide a print capability, and in some cases this can adequately replace the existing 3270 print capability. The printer would most likely be triggered by a Transient Data queue, which would need to be defined. Some method of associating a client with a corresponding printer would also need to be defined, similar to the situation with real 3270 terminals and printers.

CICS Client printers also provide a potential solution for the problem of unsolicited messages. CICS provides the ability to asynchronously schedule a transaction to a particular terminal, using the EXEC CICS START command. If the terminal is available, the specified CICS transaction will be automatically started. The transaction can issue terminal commands, such as EXEC CICS SEND MAP. This is a problem with ECI-based client programs, since there is no terminal associated with the client. CICS clients provide the ability to define a "printer". This "printer" must be defined to CICS on the host as well. When the printer is defined to the CICS client, the name of a user program must be specified. This user program is started when data is received by the client for the specified printer. The data is passed to the specified program, which must then process the data. Since this is a user specified program (which ostensibly prints the data, but really can do anything it wants with the data), the program can in fact pass the data to a client program.

Use of certain BMS options

While most applications use relatively simple BMS commands (RECEIVE MAP, SEND MAP, SEND TEXT), there are several more complicated BMS commands which are not easy to convert. These include the following options to BMS SEND:

- SEND with ACCUM option
- SEND PAGE
- BMS paging and routing

If an application uses any of the above facilities, then it is probably a poor candidate for conversion to be ECI callable, and the techniques described in this SupportPac still leave a considerable amount of manual changes. The application must be analyzed, and significant changes made to allow conversion to be ECI callable. Furthermore, it is usually much more difficult to write a new 3270 front end to the converted program.

The only thing that can be said in favor of the ECI conversion approach is that any "screen scraping" approach can also be very difficult.

The use of EXEC CICS RECEIVE statement with the BUFFER is not supported by this technique. Unlike all other 3270 terminal commands, the RECEIVE BUFFER command reads the entire contents of the 3270 buffer, without any requirement for the user to press any key. This command is therefore automatically conversational.

The conversion program will merely comment out the RECEIVE BUFFER command. This statement is rarely used, and the most common use is to save the entire contents of the screen, to allow it to be restored later. For example, if an application might want to allow a user to break out of one transaction and run a completely different application, and then return to the original application at the same point. If the application first issues a RECEIVE BUFFER, saves the result, invokes the second application, and then restores the screen contents when the second application is finished, the user can then continue from where they left off. In this case, the function to save and restore the application context might well be moved out to the new client application, and therefore no further action is really needed other than removing the RECEIVE BUFFER (as well as the EXEC CICS SEND which restores the screen). Other uses, however, may require more extensive changes, and therefore the application may not be a good candidate for conversion to be ECI callable.

Use of "terminal front end or menu programs" rather than BMS

In some cases, packages are used to handle terminal and/or menu functions. In these cases, the source program is not a BMS application. These programs are not good candidates for conversion to be ECI callable, using the techniques and tools in this SupportPac.

No source code or languages other than COBOL

The techniques and utilities provided with this package operate on CICS COBOL source programs which use BMS support. If the source programs are not available, or are not written in CICS command level COBOL using BMS, then the techniques and utilities described in this SupportPac cannot be used.

In the case of PL/I or assembler CICS programs, it is possible that many of the techniques described can be used. The utilities, however, work only with COBOL programs.

New Front Ends

Writing a new 3270 front end

Once the existing 3270 program has been converted to be called from a client rather than driven from a 3270 terminal, there is a concern about having to maintain two similar programs. It is therefore desirable to write a new 3270 front end program to drive the converted back end program. This enables the same program to be used by both 3270 and non-3270 users.

The design of the new 3270 front end program is usually fairly simple. The new program does not contain any business logic.

Basic Design

The basic design of the program is as follows:

```
RECEIVE input into COMMAREA
Move user COMMAREA into new COMMAREA, if necessary
FILL in STDHEAD fields from EIB, etc
LINK to converted program
WRITE output from COMMAREA
RETURN, leaving user COMMAREA if specified
```

A sample of such a program is included (EFHCALL.CCP).

Automatic generation of new front end

The conversion utility can generate a skeleton program, based on a template file (NEW3270.TXT). The required BMS commands, with the appropriate tailoring and options, are automatically inserted based on the BMS statements found in the original source program.

The front end should check the various switches in the standard COMMAREA header, and issue the appropriate BMS SEND command. The conversion utility inserts the appropriate SEND commands and IF statements to handle this requirement.

The conversion utility does most of the work to generate a new 3270 front end. However, there are certain things which this program cannot easily handle, and therefore the generated program will require some manual editing.

The main thing that must be handled manually is inserting COBOL copy commands for any user COMMAREA and BMS map areas used by the application. These should be inserted after the COPY STDHEAD statement that is automatically inserted. The COMMAREA layout must match the COMMAREA layout of the converted program.

The other major area that requires manually fix up is if the source program is spread out across multiple separately compiled source programs. The problem here is that if there are BMS commands in a separately compiled module, they will not be seen by the conversion utility, and hence the necessary SEND or RECEIVE options will probably not be automatically inserted.

Handling of additional parameters on BMS SEND commands

There are several optional parameters which can be specified on BMS SEND commands. These include the following BMS parameters:

- ERASE
- ERASEAUP
- MAPONLY
- DATAONLY
- FRSET
- FREEKB

Multiple output maps

If a program only uses a single map, then the new 3270 front end only needs to check for a SENDTEXT, otherwise it can issue the BMS send directly. However, if the program uses more than one map, then there must be a BMS SEND for each possible map.

The 3270 front end must check for each map name and possibly map set name, and optional parameters, and issues a BMS SEND for the appropriate map and optional parameters, from the correct data area within the COMMAREA.

3270 BMS Map Naming

There is an potential conflict with the 3270 map names. The problem arises because the levels of the generated maps start with a COBOL 01 level, and to be copied into a COMMAREA all the levels must be increased by at least one. However, the copy books may also used in the WORKING-STORAGE section in 3270 BMS applications, where they need to be at the 01 level. One solution to the problem is to increase the levels by one, but to change the 3270 program to insert a new 01 level variable with a different name, and then follow it with the copy statement. The samples use this technique, with the BMS data areas located within the new COMMAREA.

Utility Programs to Assist with Conversion

To assist with automating some of the steps in the application conversion process, three utilities have been provided as part of this SupportPac. They are all written in C. Executable versions are provided, and full source code is also provided, to allow them to be modified or tailored for individual customer environments.

COBOL Program Conversion Utility (CBLCVRT)

This program reads a CICS COBOL program and performs many of the conversion steps automatically. Input is a valid CICS COBOL program. The program produces two output files, a converted program and a messages file.

The utility takes up to three parameters. The first is the name of the COBOL program to be converted. The second is optional, and is the name of the new program. If the second parameter is omitted, the converted program will have the same file name as the original program, but with a file extension of NEW. The third parameter is also optional. It is the name of a new 3270 front end program, which will invoke the converted program. The command syntax is as follows:

```
CBLCVRT input-file  <output-file <new-3270-front-end-file>>
```

For example, to convert a CICS COBOL program named DFHCALL.CCP in the \cobol\source directory, and produce a converted program named NEWCALL and a new 3270 front end named EFHCALL, both in the \cobol\newsrce directory, the following commands could be used:

```
CD \COBOL
CBLCVRT SOURCE\DFHCALL NEWSRCE\NEWCALL NEWSRCE\EFHCALL.CCP
```

In this example, the CBLCVRT.EXE and NEW3270.TXT files should be in the \COBOL directory.

The program performs the following operations:

- If a new program name is specified, the PROGRAM-ID paragraph is changed to reflect this name.
- If no LINKAGE SECTION is found, one is inserted, as well as a DFHCOMMAREA statement and a COPY statement for a standard COMMAREA header file (STDHEAD).
- EXEC CICS ADDRESS TCTUA or COMMAREA statements are changed to COBOL SET statements, setting the address to point to the corresponding control blocks within the COMMAREA.
- BMS SEND MAP statements are changed to COBOL MOVE statements, setting the name of the BMS map in the COMMAREA header.
  - If the ERASE or ERASEAUP keyword is found, then the erase byte is set appropriately in the COMMAREA header.
  - If the MAPONLY or DATAONLY parameter is found, then the SENDTYPE field in the COMMAREA header is set appropriately.
  - Any LENGTH or FROM parameters are remembered, and used if the new 3270 front end parameter is specified.
- BMS RECEIVE MAP statements are changed to COBOL IF statements, which check that the map name specified in the COMMAREA header matches the expected map name. If any HANDLE MAPFAIL or HANDLE AID statements have been found, additional IF statements will be added to check the maplen and usraid fields in the COMMAREA header. A short routine will be added to the end of the COBOL program to handle the situation where the map names do not match.

- SEND TEXT statements will be changed to move the data into the text area field in the COMMAREA header, and set the map name to SENDTEXT.
- If BMS CONVERSE statements are found, they will be treated as a SEND statement. However, a warning message is issued. CONVERSE statements indicate that the program is conversational, and conversational programs may not be good candidates for the techniques described in this SupportPac. For further discussion, please see the Known Problems section.
- SEND CONTROL statements are commented out. No action is necessary, as this function is taken over by the new front end program.
- If CICS RETURN statements are found with either the TRANSID with or without the COMMAREA and LENGTH options, the RETURN statement is changed to a simple return. A COBOL MOVE statement is inserted, to move the TRANSID parameter to the COMMAREA header. If the COMMAREA and LENGTH options are found, then the COMMAREA length is noted in the USRCALEN field, and the COMMAREA is moved to the USRCOMMAREA field in the main COMMAREA. If no length parameter is specified, then the USRCALEN field is set to zero.
- References to several EIB fields, as well as references to DFHCOMMAREA, are changed to refer to fields within the new COMMAREA header or the USRCOMMAREA, respectively. The EIB field names which are changed are:
  - EIBTRMID - Terminal name
  - EIBTRNID - Transaction name
  - EIBAID - AID key value
  - EIBCPOS - Cursor position
  - EIBCALEN - COMMAREA length
- If any SEND ACCUM or PAGE commands are found, a warning message is written to the message log.
- If a RECEIVE BUFFER command is found, a warning message is written to the message log.
- Messages are written to the message log for any EXEC CICS LINK or XCTL statements that are found. The called programs should be checked for any 3270 dependencies.

COBOL comment statements ('*' or other special characters in column 7) are recognized and ignored.

Handling of user copy books

CICS application programs may use COPY statements to include common subroutines in a program. The COPY statements can be used in any part of the program, including the Data Division and the Procedure Division. If a program uses any CICS copy books in the Procedure Division, and the copy books contain any references to CICS terminals, then the copy books must be converted as well as the main program.

The conversion utility will generate converted versions of any copy books which are found in the current directory. Before the conversion utility is run, the main program plus any related copy books should be moved into the current directory. The utility will then check the copy books as well as the main program for any statements which must be converted. The conversion utility will write out new versions of all copy books. The converted versions will have a file extension of *new*, while the original copy books are assumed to have no file extension. If a copy book is not found, then processing will resume with the next statement in the main program and a warning message will be written to the message log.

Any statements beginning with the characters "-INC" will be treated as copy books. The CICS copy books DFHAID, DFHBMSCA and DFHEIBLK are recognized and ignored.

Generation of a new 3270 front end program

If a valid file name is specified in the third parameter, then a new 3270 COBOL front end program will be generated, using the specified file name. The generated program will include RECEIVE MAP statements for all RECEIVE MAP statements found in the program, and SEND MAP and/or SEND TEXT statements for all SEND statements found. The appropriate IF statements will be generated around the SEND statements, to execute the proper SEND statement.

The generated program will be based on a template COBOL program, which is kept in the NEW3270.TXT file. This file can be tailored to suit individual preferences and requirements. The PROGRAM-ID will be replaced with the name of the newly generated file, and the name of the called program will be replaced with the file name of the input file. The two comment lines indicating where the RECEIVE and SEND statements will be inserted will be replaced with the necessary CICS statements. These lines should not be changed.

BMS Map Source Level Changer (CHGMAP)

This is a very simple program which takes a COBOL COPY book, often a BMS generated copy book, and increases the levels of all items by one. It may also be useful for any user COMMAREA copy books. The command syntax is:

```
CHGMAP input-file <output-file>
```

If no output file name is specified, then the output file name will default to the input file name with an extension of NEW appended.

COBOL comment statements ('*' in column 7) are recognized and ignored.

For example, to change the levels of the BMS map copy book DFHCGC.CBL, producing a new copy book named EFHCGC.CBL, the following command would be used :

```
CHGMAP DFHCGC.CBL EFHCGC.CBL
```

DFHCNV Conversion Table Generator (BUILDCNV)

This utility will take a COBOL copy book and generate a corresponding set of CICS DFHCNV statements for host CICS, to provide the necessary data translation when converted CICS programs are executed on host systems and called from clients on PC systems. The input is one or more valid COBOL copy book.

Since the new CICS COMMAREA contains several distinct areas, and is likely to be made up of several copy books, the utility will accept more than one input file. The utility expects one or two parameters. The first parameter is the name of the COPY book(s). If more than one COPY book file is to be processed for a single COMMAREA, the individual file names must be separated by plus signs (+). The second parameter is the name to be used for the file containing the DFHCNV macros. For example, to process a COMMAREA consisting of four COBOL copy books, and to create a file containing the DFHCNV macros with a name of NEWCOMM.CNV, the following command could be used:

```
BUILDCNV STDHEAD+USRCOMM+DFHCGA+DFHCGB \TEST\NEWCOMM.CNV
```

A message file, with an extension of MSG, will also be created, using the same name as the DFHCNV macro file. If the second parameter is omitted, then a file name will be generated based on the file name of the first COBOL copy book specified in the first parameter, with an extension of CNV appended.

If problems are encountered

Please note that the utilities are expecting valid COBOL input. In the case of the Program Conversion Utility, this would be a complete COBOL source program. The program should compile without errors. Invalid programs may well yield unpredictable results. While some efforts have been made to recognize and ignore invalid syntax and statements, no testing has been done with incorrect input. The Program Conversion Utility expects complete programs, and not parts of programs.

The Map Source Level Changer and the Conversion Table Generator program expect valid COBOL copy book syntax.

If any problems are encountered with the utilities or documentation provided with this SupportPac, please send a note to:

macnair@us.ibm.com

If the problem is with one of the utilities, please attach the input file which is failing, as well as a description of the problem. While the author makes no promises of support, he is interested in any problems with this SupportPac.

The author is also interested in any suggestions or recommendations for improvements. If possible, please include a sample COBOL program and necessary copy books, to be used for testing purposes.

SupportPac Contents

This SupportPac is delivered as a single zip file (CA03.ZIP). The zip file contains the following files:

LICENSE2.TXT	License Agreement for Supportpac
BLDCNV.C	DFHCNV Macro generation utility (source)
BLDCNV.EXE	DFHCNV Macro generation utility
CAPTURE.CCP	Simple program to aid in debugging
CBLCVRT.C	COBOL Conversion program (source)
CBLCVRT.EXE	COBOL Conversion program
CHGMAP.C	Map level change program (source)
CHGMAP.EXE	Map level change program
DFHCBRW.CCP	Pseudo-conversational browse program
DFHCGA	COBOL COPY book for Menu map
DFHCGB	COBOL COPY book for Detail map
DFHCGC	COBOL COPY book for Browse map
EFHCBRW.CCP	New 3270 front end for browse
EFHCALL.CCP	New 3270 front end for inquiry, etc
NEW3270.TXT	COBOL Template file for CBLCVRT program
NEWCALL.CCP	Converted sample program
NEWCBRW.CCP	Converted sample program
STDHEAD	Standard COMMAREA header
USRBRWS	User COMMAREA for DFHCBRW.CCP
USRCOMM	User COMMAREA for NEWCALL.CCP
VERICOMM	User COMMAREA for VERIFYPW.CCP
VERIFYPW.CCP	Host Program to validate userid/password
VERIFYPW.TS	Modified user id validation for TS/NY

Source versions of all the programs are provided, as are executable versions of the utility programs. The utility programs are written in Microsoft Visual C and compiled with the Microsoft Visual C++ compiler V5™. The utility programs are 32-bit windows programs, and should run under either Microsoft Windows NT™ or Microsoft Windows 95™. The executable versions of the converted COBOL programs requires the IBM Transaction Server for Windows NT V4, which only runs on Windows NT. The COBOL programs have been compiled with the IBM VISUALAGE COBOL compiler V2.1. The VERIFYPW program is supplied as source only, and is intended for execution on CICS/ESA V4.1 or later. It uses CICS API statements which are not supported on the NT or AIX Transaction Server product. The CICS sample programs should be portable to other CICS server platforms. A version of the program with the unsupported APIs commented out and suitable for execution on the IBM Transaction Server for Windows NT V4 is provided as file VERIFYPW.TS. The file extension must be changed to CCP for compilation.

The original 3270 versions of the programs are shipped in the \OPT\CICS\SRC\SAMPLES\IVP directory.

No real installation of the utility programs is required. The supplied zip file should be unzipped, and the executable programs moved to a directory which is included in the Windows NT path.

Sending your comments to IBM

Changing existing CICS 3270 applications to enable new Client/Server and Web access

CA03 SCRIPT

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.
- By fax:
  - From outside the U.K., after your international access code use 44 1962 841409
  - From within the U.K., use 01962 841409
- Electronically, use the appropriate network ID:
  - IBMLink: IBMGB(TSCC)
  - Internet: tscc@hursley.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 and address/telephone number/fax number/network ID.

Readers' Comments

Changing existing CICS 3270 applications to enable new Client/Server and Web access

CA03 SCRIPT

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

Changing existing 3270 Applications
CA03 SCRIPT

---

1 Cut along this line

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

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

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.).

---

2 Fold along this line

By air mail
Par avion

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

NE PAS AFFRANCHIR
NO STAMP REQUIRED

IBM

REPONSE PAYEE
GRANDE-BRETAGNE

IBM United Kingdom Laboratories Limited
Information Development Department (MP 095)
Hursley Park
WINCHESTER, Hants
SO21 2ZZ United Kingdom

3 Fold along this line

From: Name ____________________
Company or Organization ____________________
Address ____________________
____
EMAIL ____________________
Telephone ____________________

1 Cut along this line

4 Fasten here with adhesive tape