CICS® Transaction Server for VSE/ESA™

Application Migration Aid Guide

Release 1

SC33-1943-03

CICS® Transaction Server for VSE/ESA™

Application Migration Aid Guide

Release 1

SC33-1943-03

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 25.

Fourth Edition (September 2005)

This edition applies to Release 1 of CICS Transaction Server for VSE/ESA, program number 5648-054, and to all subsequent versions, releases, and modifications until otherwise indicated in new editions. Make sure you are using the correct edition for the level of the product.

The CICS for VSE/ESA Version 2.3 edition remains applicable and current for users of CICS for VSE/ESA Version 2.3.

Order publications through your IBM representative or the IBM branch office serving your locality.

At the back of this publication is a page entitled “Sending your comments to IBM”. If you want to make any comments, please use one of the methods described there.

Contents

Preface

What this book is about

This book describes the IBM® CICS Application Migration Aid provided in CICS Transaction Server for VSE/ESA Release 1. It explains how the aid works and explains the setup and operating procedures in a VSE/ESA™ environment.

Who should read this book

This book is for intended to be used by CICS® application programmers who will use the Application Migration Aid for VSE to convert macro-level application programs to command-level application programs.

What you need to know to understand this book

You need to be familiar with writing application programs in CICS. In particular, you need either experience in macro-level programming or, at least access to the *CICS/DOS/VS 1.7 Application Programmer's Reference Manual (Macro Level)*. You also need to be familiar with the CICS/VSE 2.3 or the CICS Transaction Server for VSE/ESA Release 1 command-level application programming interface.

Book structure

Chapter 1, "Overview" gives an overview and the reasons for migrating from macro-level to command-level. It describes the programming differences between the two interfaces. An appreciation of these is necessary because the migration aid cannot convert all macros automatically—the more complex macros will require manual data input from you to complete their conversion.

Chapter 2, "The conversion process in detail" uses a series of examples to illustrate the conversion process. In particular, it gives examples of the data input required by more complex macros.

Chapter 3, "Operation" explains the procedure for running the migration aid and how to deal with the output.

Notes on terminology

The terms listed in Table 1 are commonly used in the CICS Transaction Server for VSE/ESA Release 1 library. See the *CICS Glossary* for a comprehensive definition of terminology.

Table 1 (Page 1 of 2). Commonly used words and abbreviations in CICS Transaction Server for VSE/ESA Release 1

Term	**Definition (and abbreviation if appropriate)**
$(the dollar symbol)	In the character sets and programming examples given in this book, the dollar symbol ($) is used as a national currency symbol and is assumed to be assigned the EBCDIC code point X'5B'. In some countries a different currency symbol, for example the pound symbol (£), or the yen symbol (¥), is assigned the same EBCDIC code point. In these countries, the appropriate currency symbol should be used instead of the dollar symbol.
BSM	BSM is used to indicate the basic security management supplied as part of the VSE/ESA product. It is RACROUTE-compliant, and provides the following functions: • Signon security • Transaction attach security
C	The C programming language
CICSplex	A CICSplex consists of two or more regions that are linked using CICS intercommunication facilities. Typically, a CICSplex has at least one terminal-owning region (TOR), more than one application-owning region (AOR), and may have one or more regions that own the resources accessed by the AORs
CICS Data Management Facility	The new CICS Transaction Server for VSE/ESA Release 1 facility to which all statistics and monitoring data is written, generally referred to as “DMF”
CICS/VSE	The CICS product running under the VSE/ESA operating system, frequently referred to as simply “CICS”
COBOL	The COBOL programming language
DB2 for VSE/ESA	Database 2 for VSE/ESA which was previously known as “SQL/DS”.

Table 1 (Page 2 of 2). Commonly used words and abbreviations in CICS Transaction Server for VSE/ESA Release 1

Term	Definition (and abbreviation if appropriate)
ESM	ESM is used to indicate a RACROUTE-compliant external security manager that supports some or all of the following functions: • Signon security • Transaction attach security • Resource security • Command security • Non-terminal security • Surrogate user security • MRO/ISC security (MRO, LU6.1 or LU6.2) • FEPI security.
FOR (file-owning region)—also known as a DOR (data-owning region)	A CICS region whose primary purpose is to manage VSAM and DAM files, and VSAM data tables, through function provided by the CICS file control program.
IBM C for VSE/ESA	The Language Environment version of the C programming language compiler. Generally referred to as “C/VSE”.
IBM COBOL for VSE/ESA	The Language Environment version of the COBOL programming language compiler. Generally referred to as “COBOL/VSE”.
IBM PL/I for VSE/ESA	The Language Environment version of the PL/I programming language compiler. Generally referred to as “PL/I VSE”.
IBM Language Environment for VSE/ESA	The common runtime interface for all LE-conforming languages. Generally referred to as “LE/VSE”.
PL/I	The PL/I programming language
VSE/POWER	Priority Output Writers Execution processors and input Readers. The VSE/ESA spooling subsystem which is exploited by the report controller.
VSE/ESA System Authorization Facility	The new VSE facility which enables the new security mechanisms in CICS TS for VSE/ESA R1, generally referred to as “SAF”
VSE/ESA Central Functions component	The new name for the VSE Advanced Function (AF) component
VSE/VTAM	“VTAM”

Determining if a publication is current

IBM regularly updates its publications with new and changed information. When first published, both the printed hardcopy and the BookManager softcopy versions of a publication are in step, but subsequent updates are normally made available in softcopy before they appear in hardcopy.

For CICS Transaction Server for VSE/ESA Release 1 books, softcopy updates appear regularly on the *Transaction Processing and Data Collection Kit* CD-ROM, SK2T-0730-xx and on the *VSE/ESA Collection Kit* CD-ROM, SK2T-0060-xx. Each reissue of the collection kit is indicated by an updated order number suffix (the -xx part). For example, collection kit SK2T-0730-20 is more up-to-date than SK2T-0730-19. The collection kit is also clearly dated on the front cover.

For individual books, the suffix number is incremented each time it is updated, so a publication with order number SC33-0667-02 is more recent than one with order number SC33-0667-01. Updates in the softcopy are clearly marked by revision codes (usually a “#” character) to the left of the changes.

Note that book suffix numbers are updated as a product moves from release to release, as well as for updates within a given release. Also, the date in the edition notice is not changed until the hardcopy is reissued.

Chapter 1. Overview

This chapter

- Introduces the CICS Application Migration Aid, and describes why conversion to command-level is needed
- Lists the differences between the macro-level and command-level APIs

The CICS Application Migration Aid is designed to assist you in converting CICS application programs from the macro-level API to the command-level API. Applications written in assembler, COBOL, or PL/I can be used with the migration aid.

It converts the simpler macros in an application program automatically. It partially converts the more complex macros, issuing messages that guide you to complete the conversion.

The migration aid executes as an offline batch utility. It runs on any VSE/ESA system.

Why convert?

The macro-level and command-level interfaces are different ways in which an application can communicate with CICS. The decision to convert from the use of macro-level to the use of command-level can be based on the following considerations:

- **Command-level is easier to use**

  It is a high-level language that gives you improved programmer productivity because it is easier to write, debug, maintain, and extend. By avoiding dependence on CICS control blocks, it increases reliability and release-to-release compatibility for application programs.

- **Command-level has more function**

  You may wish to extend an application, and need functions not available at the macro-level, particularly intersystem communication and multiregion operation. It is easier to split CICS regions for multiregion operation—command-level API supports function shipping, macro-level does not.

- **Command-level offers you virtual storage constraint relief**

  Application programs may need to be moved above the 16MB line, where macro-level programs cannot reside.

- **The macro-level programming interface is no longer supported**

  To build applications for this and any future releases of CICS Transaction Server for VSE/ESA, you must use the command-level API.

What's the difference?

Compared to a programming language, an API is much simpler. It has three major components—language, data, and condition handling. The CICS command-level API differs from the macro-level in all three components.

1. **Language**

   The command language is more natural, more explicit, more concise, and less dependent on context. It is not always easy to look at a macro and tell what it does.

2. **Data**

   In macro-level programs, the way in which data is passed between an application and CICS has caused problems in the past, and will cause most difficulty in conversion. There are two problems—elements of data are not well separated, and the storage where they reside tends to belong to CICS rather than to the application. Programs are difficult to write and read because you need to know about data structures and the lifetime of storage. It is very easy to overwrite CICS internal data and cause CICS to abend.

   In command-level programs, data elements are separated from each other and from CICS data; responsibility for working storage lies largely with the program itself. What happens to data is therefore more visible.

3. **Condition handling**

   Condition handling in macro-level programs is specified locally, that is, at or near the macro itself. In some ways this is a good thing, but it means that there is no facility for specifying a standard system action that allows you to cope with conditions that should never arise (but might) without having to write any code.

   The command-level interface provides you with a choice of global or local condition handling. The migration aid converts the local condition handling in the macro-level source to local condition handling in the command-level output.

How the migration aid works in the conversion process

The conversion procedure is shown in Figure 1.

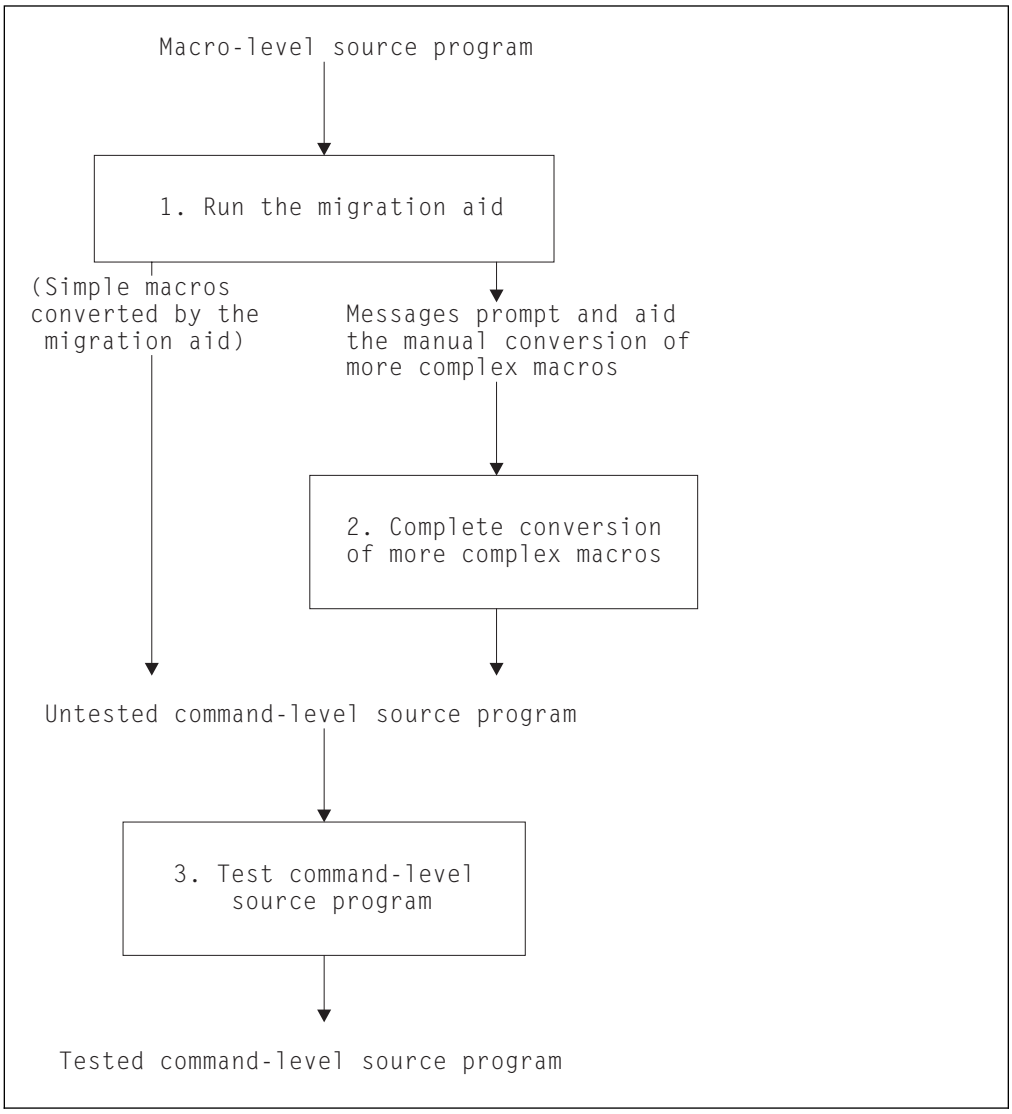

Figure 1. The conversion procedure. The boxes show the three stages in the procedure.

The migration aid takes a macro-level source program as its input and deals with the macros in the program in one of two ways:

1. The migration aid converts the simpler macros in the program to the equivalent command-level functions. A program that contains only this sort of macro is entirely converted by the migration aid. However, most programs also contain more complex macros.
2. The migration aid cannot convert these complex macros; instead, it starts the conversion for each macro and issues a message prompting you to supply missing data to complete the conversion.

The output is written to a dataset and, in most cases, does not even assemble or compile successfully without manual intervention. In any case, the command-level program will require thorough testing before it can be put into production. The input to the migration aid must be a debugged, functioning, application program.

Which macros are converted?

The migration aid converts those macros documented in the *Application Programmer's Reference Manual (Macro Level)* except for DFHFC TYPE=DL/I macros. They correspond to EXEC DLI rather than EXEC CICS commands, and are not converted.

A great many macro functions exist in CICS and are not documented in the *CICS/DOS/VS 1.7 Application Programmer's Reference Manual (Macro Level)*. These are not part of the API and your application should not use them, although many do. The migration aid will advise you to consult your system programmer.

Software requirements

You can run the CICS Application Migration Aid under VSE/ESA. (It does not run under VM.) You install it using normal PTF installation procedures, either directly with MSHP or via the VSE/ESA interactive interface dialogs.

Chapter 2. The conversion process in detail

This chapter

- Illustrates the conversion process with examples. These include descriptions of how you set about manually completing conversions for more complex macros.
- Describes condition handling and control blocks.
- Describes what needs to be done after the conversion of the macros and lists some special assembler considerations.

The examples in this chapter include the macro-level commands for assembler programs. For COBOL programs, the output command would be the same, but followed by `END-EXEC`. For PL/I programs, the output command would also be the same, but followed by a semi-colon (;).

Example 1

Macro

```
DFHTD TYPE=PURGE,DESTID=CSML
```

is converted to:

Command

```
EXEC CICS DELETEQ TD QUEUE('CSML')
```

Our first example is a simple case of a macro that converts directly to a command, and illustrates the difference in the language of the two interfaces. The migration aid can perform such conversions for you.

This saves you some typing and tedious consultation of manuals, but you should not dispense with the manuals altogether. You need to know what function the code actually performs. In this case, it is to delete all the transient data associated with a particular intrapartition destination (or queue), CSML, and to free the associated storage.

Example 2

Macro

```
DFHTD TYPE=PURGE
```

is converted to:

Command

```
EXEC CICS DELETEQ TD QUEUE(TCATDDI)
*ERCM0P02 04 - QUEUE NAME ASSUMED TO BE IN TCATDDI
```

This example is a simple variation on the first one. It shows how the storage for variables is owned by CICS in the macro interface, but is owned by the program in the command-level interface.

Throughout the macro interface, it is possible to specify values either on the macro itself or by placing it in a field in the task control area (TCA) prior to executing the macro. In fact, the latter method is the only way of specifying variable data.

In the present example, the TCA field from which CICS obtains the name of the queue is `TCATDDI`. This is typical of how hard to remember such field names can be and, here again, the migration aid will save you some tedious, error-prone reading, and typing.

The example is a correct conversion, but it is not complete because a command-level program does not have access to the TCA at all. If a macro-level program contains the CICS-supplied definition of the TCA, this should be removed. To complete the conversion in this case, a declaration is required for `TCATDDI`, which is four characters, in your working storage. The migration aid provides a message to this effect.

Next it is advisable to change the name of the field to something more sensible than `TCATDDI`, to indicate both what the data is for and that it is not in the TCA. Of course, you must change all references to it as well. Such renaming is not usually essential but it has benefits.

You will learn more about what the program does, and you will make it easier to read.

You may find that the macro is always executed with the same value, so the data is not actually a variable. In this case you can code a constant in the command and remove the relevant references to the field.

You may find that the field is being set in another program, in which case replacing the TCA field with one in working storage fails. One of the great dangers of the macro-level API is that it encouraged the passing of data between programs in CICS control blocks, which obscures parts of the program interfaces.

Example 3

Macro

```
DFHTD TYPE=PUT,DESTID=CSML,TDADDR=TDOAVRL
```

is converted to:

Command

```
EXEC CICS WRITEQ TD QUEUE('CSML') FROM(????) LENGTH(????)
*ERCM0P03 08 - FROM() OPTION REQUIRED
*ERCM0P04 08 - LENGTH() VALUE REQUIRED
```

This example illustrates the lack of data separation in the macro interface. It shows, partially, how to direct transient data to a queue. Usually, you have to

provide an area that contains both the data and its length in a particular format. Matters are further complicated by:

- The fact that the length includes the length of the length field.
- In some cases, you don't specify the length at all.
- The CICS copybooks for this and the next example include other fields that precede the data.

Not all macros cause this much trouble. DFHTD and DFHTS can be difficult.

The command equivalent is relatively simple. The FROM value specifies the data to be written and the LENGTH value specifies its length.

Here are the message explanations provided by the migration aid:

```
*ERCM0P03 08 - FROM() OPTION REQUIRED
```

The TDADDR parameter of the DFHTD TYPE=PUT macro specifies the storage area containing the data to be written. If it is omitted, the address of the storage area is assumed to be in TCATDAA. For intrapartition data and variable-length extrapartition data, the first four bytes of the storage area must contain the length in LLbb format. For fixed-length extrapartition data, the storage area contains only the data. The FROM option of the WRITEQ TD command specifies only the data in all cases. If a program uses the DFHTDOA copybook, the data will probably be whatever immediately follows it.

```
*ERCM0P04 08 - LENGTH() VALUE REQUIRED
```

For fixed-length extrapartition data, the DFHTD TYPE=PUT macro does not specify the length of the output data because CICS knows it from the destination control table (DCT) entry. For intrapartition data and variable-length extrapartition data, the length is specified at the beginning of the output area in LLbb format, **including 4** for the length field itself. If an application program uses the DFHTDOA copybook, the length is probably set in a reference to TDOAVRL. The LENGTH option of the WRITEQ TD command should specify only the length of the data in all cases. You may be able to specify a constant or you may need to put the length in a halfword field in your working storage and specify the name of that halfword in the LENGTH option.

Example 4

Macro

```
DFHTD TYPE=GET,DESTID=CSML
```

is converted to:

Command

```
EXEC CICS READQ TD QUEUE('CSML') SET(????) LENGTH(????)
*ERCM0P05 08 - SET() OPTION REQUIRED
*ERCM0P06 08 - LENGTH() FIELD MAY BE REQUIRED
```

Finally, we have an example showing retrieval of transient data. The migration aid provides the following explanation:

```
*ERCM0P05 08 - SET() OPTION REQUIRED
```

The DFHTD TYPE=GET macro returns the address of the data retrieved at TCATDAA. For intrapartition data and variable-length extrapartition data, the first four bytes at this address contain the length in LLbb format. For fixed-length extrapartition data, the address points to the start of the data. If an application program uses the DFHTDIA copybook, it probably loads TDIABAR from TCATDAA and then adjusts it to allow for data at the start of the copybook. All of this should be removed. In the READQ TD command, the SET option always returns the address of the data.

```
*ERCM0P06 08 - LENGTH() FIELD MAY BE REQUIRED
```

For fixed-length extrapartition data, the DFHTD TYPE=GET macro does not return a length because the program will know what it is. In this case, you should not need to specify LENGTH at all. For intrapartition data and variable-length extrapartition data, the length is returned at the beginning of the output area in LLbb format, **including 4** for the length field itself. If an application program DFHTDIA copybook, the length is probably referred to TDIAIRL. The LENGTH option of the READQ TD command should specify a halfword in your working storage. Don't forget that the command always returns the actual length of the data.

Condition handling

Macro

```
          DFHTD TYPE=GET,NORESP=label
```

or

```
          DFHTD TYPE=CHECK,NORESP=label
```

or

```
          CLI  TCATDTR,X'00'
          BE   label
```

If an error occurs in the execution of a macro, control returns to the program, which can deal with the error in one of three ways (or not at all):

1. Operands can be coded on the macro, specifying a label to which control passes if execution of the macro causes the specified condition. (One of the possible conditions is normal response.)
2. The macro can be followed by a TYPE=CHECK macro, specifying conditions and labels in same way.
3. Finally, the macro can be followed by a test of a response code in a control block, usually the TCA. Transient data response codes are in TCATDTR for assembler and PL/I, and TCATDRC for COBOL.

The migration aid converts the first two of these to tests of EIBRESP and appropriate branches. You must deal with the third case.

Command-level default condition handling does not cause control to return to the application. To override this, an EXEC CICS IGNORE CONDITION ERROR command must be executed at the start of the program.

If the program contains a great many error tests, you may wish to use HANDLE CONDITION instead. For information about the HANDLE CONDITION command, see the appropriate programming reference manual:

- For CICS/DOS/VS, see the *Application Programmer's Reference (Command Level)* manual.
- For CICS/VSE, see the *Application Programming Reference* manual.

Control blocks

The migration aid converts macro language to command language, advise you of fields that you need in your working storage, and indicate how you need to change references to data structures into references to data itself.

You must ensure that the declarations for the common system area (CSA), task control area (TCA), and terminal control table terminal entry (TCTTE) are removed, because a command-level program cannot access these areas. These declarations may be have been used by a macro-level program to address the common work area (CWA), transaction work area (TWA), or terminal control table user area (TCTUA) respectively. If this is the case, you will need to add the appropriate EXEC CICS ADDRESS commands to the converted program.

In the process of amending references to data structures, you will also have removed declarations such as DFHTDOA. In this way, references to CICS control blocks that were needed to make the macros work can be removed. However, the program might still refer to some other fields in CICS control blocks, which it can no longer access. Deal with these references by using ASSIGN commands, the EXEC interface block, or INQUIRE/SET commands, as described below.

Using ASSIGN commands

Some control block data is available to command-level programs by means of the ASSIGN command or via the SPI. Most of the values you can access in this way are associated with the application's terminal, and correspond directly with TCTTE fields. See Appendix A, "Control block fields—ASSIGN options" on page 17 for more details about equivalent methods.

A reference in a macro program to TCTTEOI, for instance, could be converted by including an EXEC CICS ASSIGN OPID(TCTTEOI) command, together with a declaration naming the field (still referred to as TCTTEOI) in working storage as 3 characters. As before, it is advisable to use a different name, and also change all the references. If none of your fields are supported by the API or SPI, you should change the logic to eliminate the references.

Data in bit form is also available. EXEC CICS ASSIGN EXTDS essentially returns the value of the bit TCTTEEDS, but as a whole byte, so the references have to be changed anyway. Appendix A, "Control block fields—ASSIGN options" on page 17 lists the most commonly referenced control block fields and flags and the appropriate ASSIGN option that may be used instead. Because the ASSIGN command has been extended with each release of CICS, you should refer to the

appropriate *Application Programmer's Reference* manual to check that the option is supported for the version of CICS in use.

Using the EXEC interface block

The command-level interface has a small control block of its own called the EXEC interface block (EIB). It contains time and date fields and responses, as well as three important values that correspond to the macro-level field names TCTTETI, TCTTEAID, and TCTTECAD. References to these can be replaced directly with references to EIBTRMID, EIBAID, and EIBCPOSN respectively. References to response codes, such as TCATDTR, must be converted to tests of EIBRESP.

Using INQUIRE/SET commands

Control block data not available in one of these ways is regarded as being outside the API. So are macros that are not documented in the *CICS/DOS/VS 1.7 Application Programmer's Reference Manual (Macro Level)*, such as the CTYPE macros. Instead, these are seen as system programming functions.

A special group of EXEC CICS commands, mostly INQUIRE and SET, is provided for use by system programmers. To a large extent, programs that go beyond the API (that is, programs that use functions that are not documented in the *Application Programmer's Reference Manual (Command Level)*) can be converted to use these commands. For further details, refer to the:

- *Customization Guide* (for CICS/DOS/VS 1.7)
- *System Programming Reference* manual (for CICS/VSE Version 2)

Other considerations

The first RECEIVE

Macro programs that start from a terminal access the initial input by finding the address of a terminal input-output area (TIOA) from a field, TCTTEDA. This field is itself usually accessed by means of a load of TCTTEAR from TCAFCAAA. In order to find the data in a command-level program, the program must issue EXEC CICS RECEIVE.

Thus, what is usually two lines of simple code in a macro-level program must be replaced with a command. This can also gives rise to the next difficulty.

Front-ending

It is a common practice to group a number of terminal applications together and "front-end" them. The front-end program looks at the initial TIOA, in some cases modifies it, and then having decided which application is required, LINKs or XCTLs to it. The invoked program need not know whether it has received control from the front-end or directly from the terminal.

In command-level, the front-end must issue a RECEIVE to look at the initial input. This causes CICS to pass the data to the application and free the TIOA, and any subsequent RECEIVE waits for new input. The called program can now no longer access the initial input, whether it is a macro-level program or command-level. In fact, the correct way to do this in command-level is for the front-end to pass the necessary data to the called programs using COMMAREA. This requires the called program to be command-level, and for it to have code that is sensitive to whether it has been invoked from a program or a terminal.

Converting such a suite of applications would be difficult even if all the programs belonged to the user, but it is common for the called programs to be supplied by vendors and even by IBM. Because of these difficulties, an option INPUTMSG was added to the EXEC CICS LINK and XCTL commands in CICS/VSE Version 2 Release 2. The front-end can put whatever it likes in INPUTMSG, and the called program obtains that data either by means of its first RECEIVE (or by finding TCTTEDA, in the case of a macro-level program) just as if it were invoked directly.

Communication between programs

The front-end problem is the most severe and commonplace case of a general problem alluded to earlier. In macro-level programs, data can be passed from one program to another in CICS control blocks. In the front-end case, the data is passed in TCTTEDA. Another field sometimes passed in this way is TCANXTID, which is used by DFHPC RETURN to invoke the next transaction.

Probably the most difficult part of a conversion is detecting such cases. When you find them, you have to convert all the programs involved together. Apart from the front-end case, you have to decide on another way of passing and receiving the data. COMMAREA and the TWA are the obvious ways.

Missing functions

A few functions of the macro-level API have no command equivalents. The migration aid will advise you what to do when it detects one of these in a program.

DFHFC TYPE=DL/I macros are not converted. They correspond to EXEC DLI rather than EXEC CICS commands.

A great many macro functions exist in CICS and are not documented in the *CICS/DOS/VS 1.7 Application Programmer's Reference Manual (Macro Level)*. These are not part of the API and your application should not use them, although many do. The migration aid will advise you to consult your system programmer.

Special assembler considerations

An assembler command-level language usually needs at least two special "command-level macros"—DFHEISTG and DFHEIENT. If a program does not have them, the command language translator may insert them. See the appropriate *Application Programmer's Reference* manual for information about these.

DFHEISTG defines your working storage. This is where you redefine data that, in the macro-level program, resided in a CICS control block, usually the TCA.

DFHEIENT acquires your working storage and establishes addressability to it, as well as to the code and to the EIB. You usually need to remove user code that establishes base registers. You also usually need to find registers to use as bases for working storage and the EIB. The CSA, TCA, and TCTTE are no longer addressed; the registers that used to address them will often now be available. On the other hand, if the program accesses the CWA, TWA, or TCTUA, three registers are used to address these. Furthermore, command-level applications use R0, R1, R14, and R15 whereas, in a macro application, R15 is generally available to the program.

Finally, you should be aware that many macro-level assembler programs do not always use CICS-supplied definitions to address data. In "Example 4" on page 7,

where the use of DFHTDIA is very cumbersome, after a GET, it is easier just to load a register from TCATDAA, load a halfword from there to get the length + 4, and then add 4 to the original register to address the data. Another commonplace example is not to include DFHCSADS and access the CWA by adding 512 to R13. Such techniques will make the task of finding references to CICS control blocks more difficult for you.

Chapter 3. Operation

This chapter describes how to run the migration aid, and how to deal with the resulting output.

Running the migration aid

Macro-level source used as input to the migration aid must assemble or compile without errors, otherwise the results of conversion are unpredictable. You should therefore check that your input source is error-free. The source may be a complete program or a program segment.

An example of the JCL procedure used to run the Application Migration Aid is shown in Figure 2. It should only be modified to define your procedure library.

An example of the JCL to run the Application Migration Aid is shown in Figure 3 on page 14.

```
// JOB CATALOG AMA EXECUTION PROCEDURE
// EXEC LIBR
ACC S=proclib.sublib                          1
CAT ERCAPR01.PROC,REP=YES
// PROC CAPS=,                                2
               LANG=ASM,MSGS=MSGS,            3 4
               SRCLIB=,SOURCE=,                                    *
               NEWNAME=,                                           *
               REPLACE=
*
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                                   *
* This JCL processes macro-level source from a member defined by the  *
* parameter  SOURCE  in a library defined either  by  the  SRCLIB     *
* parameter, or the JOB SOURCE library search chain.                  *
*                                                                   *
* The command-level output is written as  a member of the same type   *
* to the library SRCLIB.  The member name is  assigned by the         *
* NEWNAME symbolic parameter.  Use the  REPLACE option if an output   *
* member of that name already exists and you want to replace it.      *
*                                                                   *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
// EXEC PGM=ERCMP0M0,                                               *
               SIZE=AUTO,                                          *
               PARM='&CAPS,&LANG,&MSGS,NEW=&NEWNAME,&REPLACE,      *
               SRCLIB=&SRCLIB,SOURCE=&SOURCE'
*
/+
/*
```

Figure 2. Example JCL procedure to run the Application Migration Aid

```
// JOB ERCRUN
*  JOB TO RUN AMA
* ---------------------------------------------------------------------
// OPTION LOG
// DLBL CICSUCT,'vsam.catalog',,VSAM
// DLBL MESSGES,'CICSERC.MESSAGES',,VSAM,CAT=CICSUCT      5
// LIBDEF *,SEARCH=(lib.sublib,PRD2.GEN1,PRD1.BASE),TEMP  6
// LIBDEF DUMP,CATALOG=SYSDUMP.sublib                     7
// OPTION SYSDUMP
// EXEC PROC=ERCAPR01,
               SRCLIB='lib.sublib',                       8     *
               SOURCE='member.type',                      9     *
               REPLACE=REPLACE,                           10    *
               NEWNAME=newmem                             11
/*
/&
```

Figure 3. Example JCL to run the Application Migration Aid

Many of the other parameters listed below are unlikely to be changed for each run. The note numbers refer to the note numbers displayed in reverse print in Figure 2 on page 13 **and** in Figure 3.

Parameters to be set:

1 Set `proclib.sublib` to the library in which you want the procedure to reside.

2 Set `CAPS=CAPS` to print message explanation text in upper case (implies `MSGS`). Set this if you are using an uppercase-only printer (for example, with Japanese Katakana characters).

3 Set `LANG=ASM`, `LANG=PLI`, or `LANG=COBOL`, depending on the type of source code to be analyzed. If you have source code in more than one language, you need a separate procedure for each.

4 Set `MSGS=MSGS` to print the text of the explanations for all messages that were produced by the conversion on SYSLST. Specify `MSGS=NO` if you do not require the explanatory text to be printed. You may want to specify `MSGS=NO` if you have run the optional job to list all the messages.

5 The default message data set name is given as `CICSERC.MESSAGES`. Change this if you changed it when you defined the message data set.

6 The library search chain should include the library containing the migration aid together with all the libraries which you would use to assemble or compile the macro-level programs. It is possible for different LIBDEF chains to be used. The migration aid uses the `// LIBDEF SOURCE...` definition in its search for all COPY files and macros, and to find the source if it is not present on the library defined by the symbolic parameter, SRCLIB.

7 Define SYSDUMP appropriately.

8 Set `SRCLIB` to the library to which the command-level source output is to be written. This library is also the first to be searched for your source.

9 Set `SOURCE` to the name of the member of the library containing the macro-level source program to be converted. If '.type' is not supplied, then it defaults to '.A' for assembler, '.C' for COBOL, or '.P' for PL/I.

10 Set `REPLACE=REPLACE` to overwrite an existing member with the same name as the output member name specified. This should not be used to replace the macro-level source member with the command-level source member, but it will do that if the input member name and library are the same as the output member name and library.

11 Set `NEWNAME` to the member name to be assigned to the generated command-level program. The type will be the same as the SOURCE.

Dealing with the results

The migration aid messages resulting from the job run are embedded as comments in the (partially) converted output source.

To deal with the messages, you should:

1. Use an editor to search for messages in the partially converted source. Each message is preceded by a number `ERCMnann` (where each `n` is a number, 0 through 9, and `a` is a letter). You can locate each message in the code by searching for `ERCM`.
2. Deal with each message by:
   - Checking any assumptions indicated by the message. The migration aid sometimes has to assume certain facts in order to carry out the conversion of a macro.
   - Entering any data requested by the message.
   - Responding to anything else in the message.

   Refer to the accompanying help text for the message. Every message has a detailed explanation of the situation that caused it to be issued and the action required. You can find this help text either:
   - From the SYSLST output, which is generated after the conversion program executes if `MSGS=MSGS` has been set in the run time job, or
   - In the listing of the message data set, if you have previously printed it.

In the output source program, messages are generated at the following levels:

Level 0 These messages provide information for you.

Level 4 These messages give you the option of doing something or not, or tell you that the conversion program has made an assumption that you should verify.

Level 8 These messages require you to take some action before the program works.

Level 12 These messages mean that the conversion program detected a macro that could not be converted.

Levels 16 or 20
These messages indicate serious problems with the conversion program—it probably will not (or did not) work at all. The return code of the jobstep is equal to the highest level of message issued.

Thorough testing after conversion is essential—there is not a 100% correspondence between macros and commands and there might be differences in

function. The conversion program might issue messages indicating that it cannot find things that are obvious to you. It follows strict search criteria and stops looking for things when there is a possibility that it would be making an invalid assumption. It does not attempt to analyze the program's logic flow.

Note that in the output source program, the sequence numbers in columns 73 through 80 are reallocated.

Conversion statistics

The results of the conversion process are summarized in the form of a table at the end of SYSLST. An example is shown below.

```
                    Not         Partially
          Converted Converted   Converted   Deleted   Total
DFHBIF        0         0           0          0        0
DFHBMS        0         0           3          0        3
DFHDC         2         0           0          0        2
DFHDI         0         0           0          0        0
DFHFC         0         0           5          0        5
DFHIC         0         0           0          0        0
DFHJC         0         0           0          0        0
DFHKC         0         0           0          0        0
DFHOC         0         0           0          0        0
DFHPC         0         0           1          0        1
DFHSC         0         0           2          1        3
DFHSP         0         0           0          0        0
DFHTC         0         0           4          0        4
DFHTD         0         0           0          0        0
DFHTR         2         0           0          0        2
DFHTS         0         0           0          0        0
Totals        4         0          15          1       20
              20%        0%         75%         5%
```

The table shows how many occurrences of each macro in the source program have been converted, not converted, partially converted, or deleted.

You need to deal with macros as follows:

Converted No further action required.

Not Converted You must convert these macros manually.

Partially Converted
You must complete the conversion of these macros manually.

Deleted No further action required. You also get an information message reporting the deletion.

All converted macros are retained in the output source as comments.

Performance

The CICS/VSE version of the migration aid has not been designed with high performance in mind. Because the migration aid uses the Librarian interface to search for COPYBOOKS and VSE macros, searches can be time-consuming. As a rough guide, it may take 10 minutes to convert a thousand lines of code (KLOC).

Appendix A. Control block fields—ASSIGN options

This appendix lists the most commonly referenced control block fields and flags, and the appropriate ASSIGN option that may be used instead. Because the ASSIGN command has been extended with each release of CICS, you should refer to the appropriate *Application Programmer's Reference* manual to check that the option is supported for your version of CICS.

Table 2 (Page 1 of 2). Control block fields and flags, with equivalent ASSIGN options.

Control block		ASSIGN option
Field name	**Flag name**	
TCAATTSC		STARTCODE
TCABMMC		MAPCOLUMN
TCABMMH		MAPHEIGHT
TCABMML		MAPLINE
TCABMMW		MAPWIDTH
TCADBRTS		RESTART
TCADSTID		QNAME
TCAFCAAA		FACILITY
TCAFCI		FCI
TCAMSLDC		LDCNUM
TCAORABC		ORGABCODE
TCTEASCC		SCRNWD
TCTEASCL		SCRNHT
TCTECSG1		GCHARS
TCTECSG2		GCODES
TCTEDSCC		SCRNWD
TCTEDSCL		SCRNHT
TCTEGMMF	TCTEGMMI	GMMI
TCTEMOP	TCTEMOPU	UNATTEND
TCTENNAM		NETNAME
TCTESIDI		SIGDATA
TCTETDST	TCTESCSB	DSSCS
	TCTETTSI	DS3270
TCTETXTF	TCTEAPKB	APLKYBD
	TCTEAPTX	APLTEXT
	TCTE327E	EWASUPP
	TCTETXT6	KATAKANA
	TCTETXKB	TEXTKYBD
	TCTETXPR	TEXTPRINT

Table 2 (Page 2 of 2). Control block fields and flags, with equivalent ASSIGN options.

Control block		ASSIGN option
Field name	**Flag name**	
TCTE32EF	TCTTECOL	COLOR
	TCTTEEDS	EXTDS
	TCTTEHIL	HILIGHT
	TCTTEMSR	MSRCONTROL
	TCTTEPRN	PARTNS
	TCTTEPSS	PS
	TCTTEVAL	VALIDATION
TCTE32E2	TCTTEBTR	BTRANS
	TCTTEFRL	OUTLINE
	TCTTEMIX	SOSI
TCTTEDLM		DELIMITER
TCTTEOCL		OPCLASS
TCTTEOI		OPID
TCTTESID		STATIONID
TCTTETAB		NUMTAB
TCTTETC		NEXTTRANSID
TCTTETI		PRINSYSID
		SYSID
TCTTETID		TELLERID
TCTTETP		TERMPRIORITY
TCTTETT		TERMCODE
TCTVAPPN		APPLID

Bibliography

CICS Transaction Server for VSE/ESA Release 1 library

Evaluation and planning	
Enhancements Guide	GC34-5763
Release Guide	GC33-1645
Migration Guide	GC33-1646
Report Controller Planning Guide	SC33-1941
General	
Master Index	SC33-1648
Trace Entries	SX33-6108
User's Handbook	SX33-6101
Glossary (softcopy only)	GC33-1649
Administration	
System Definition Guide	SC33-1651
Customization Guide	SC33-1652
Resource Definition Guide	SC33-1653
Operations and Utilities Guide	SC33-1654
CICS-Supplied Transactions	SC33-1655
Programming	
Application Programming Guide	SC33-1657
Application Programming Reference	SC33-1658
Sample Applications Guide	SC33-1713
Application Migration Aid Guide	SC33-1943
System Programming Reference	SC33-1659
Distributed Transaction Programming Guide	SC33-1661
Front End Programming Interface User's Guide	SC33-1662
REXX Guide	SC34-5764
Diagnosis	
Problem Determination Guide	GC33-1663
Messages and Codes Vol 3 (softcopy only)	SC33-6799
Diagnosis Reference	LY33-6085
Data Areas	LY33-6086
Supplementary Data Areas	LY33-6087
Communication	
Intercommunication Guide	SC33-1665
Internet Guide	SC34-5765
CICS Family: Interproduct Communication	SC33-0824
CICS Family: Communicating from CICS on System/390	SC33-1697
Special topics	
Recovery and Restart Guide	SC33-1666
Performance Guide	SC33-1667
Shared Data Tables Guide	SC33-1668
Security Guide	SC33-1942
External Interfaces Guide	SC33-1669
XRF Guide	SC33-1671
Report Controller User's Guide	SC34-5688
CICS Clients	
CICS Clients: Administration	SC33-1792
CICS Universal Clients Version 3 for OS/2: Administration	SC34-5450
CICS Universal Clients Version 3 for Windows: Administration	SC34-5449
CICS Universal Clients Version 3 for AIX: Administration	SC34-5348
CICS Universal Clients Version 3 for Solaris: Administration	SC34-5451
CICS Family: OO programming in C++ for CICS Clients	SC33-1923
CICS Family: OO programming in BASIC for CICS Clients	SC33-1671
CICS Family: Client/Server Programming	SC33-1435
CICS Transaction Gateway Version 3: Administration	SC34-5448

Books from VSE/ESA 2.5 base program libraries

VSE/ESA Version 2 Release 5

Book title	Order number
Administration	SC33-6705
Diagnosis Tools	SC33-6614
Extended Addressability	SC33-6621
Guide for Solving Problems	SC33-6710
Guide to System Functions	SC33-6711
Installation	SC33-6704
Licensed Program Specification	GC33-6700
Messages and Codes Volume 1	SC33-6796
Messages and Codes Volume 2	SC33-6798
Messages and Codes Volume 3	SC33-6799
Networking Support	SC33-6708
Operation	SC33-6706
Planning	SC33-6703
Programming and Workstation Guide	SC33-6709
System Control Statements	SC33-6713
System Macro Reference	SC33-6716
System Macro User's Guide	SC33-6715
System Upgrade and Service	SC33-6702
System Utilities	SC33-6717
TCP/IP User's Guide	SC33-6601
Turbo Dispatcher Guide and Reference	SC33-6797
Unattended Node Support	SC33-6712

High-Level Assembler Language (HLASM)

Book title	Order number
General Information	GC26-8261
Installation and Customization Guide	SC26-8263
Language Reference	SC26-8265
Programmer's Guide	SC26-8264

Language Environment for VSE/ESA (LE/VSE)

Book title	Order number
C Run-Time Library Reference	SC33-6689
C Run-Time Programming Guide	SC33-6688
Concepts Guide	GC33-6680
Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798
Debug Tool for VSE/ESA User's Guide and Reference	SC26-8797
Debugging Guide and Run-Time Messages	SC33-6681
Diagnosis Guide	SC26-8060
Fact Sheet	GC33-6679
Installation and Customization Guide	SC33-6682
LE/VSE Enhancements	SC33-6778
Licensed Program Specification	GC33-6683
Programming Guide	SC33-6684
Programming Reference	SC33-6685
Run-Time Migration Guide	SC33-6687
Writing Interlanguage Communication Applications	SC33-6686

VSE/ICCF

Book title	Order number
Adminstration and Operations	SC33-6738
User's Guide	SC33-6739

VSE/POWER

Book title	Order number
Administration and Operation	SC33-6733
Application Programming	SC33-6736
Networking Guide	SC33-6735
Remote Job Entry User's Guide	SC33-6734

VSE/VSAM

Book title	Order number
Commands	SC33-6731
User's Guide and Application Programming	SC33-6732

VTAM for VSE/ESA

Book title	Order number
Customization	LY43-0063
Diagnosis	LY43-0065
Data Areas	LY43-0104
Messages and Codes	SC31-6493
Migration Guide	GC31-8072
Network Implementation Guide	SC31-6494
Operation	SC31-6495
Overview	GC31-8114
Programming	SC31-6496
Programming for LU6.2	SC31-6497
Release Guide	GC31-8090
Resource Definition Reference	SC31-6498

Books from VSE/ESA 2.5 optional program libraries

C for VSE/ESA (C/VSE)

Book title	Order number
C Run-Time Library Reference	SC33-6689
C Run-Time Programming Guide	SC33-6688
Diagnosis Guide	GC09-2426
Installation and Customization Guide	GC09-2422
Language Reference	SC09-2425
Licensed Program Specification	GC09-2421
Migration Guide	SC09-2423
User’s Guide	SC09-2424

COBOL for VSE/ESA (COBOL/VSE)

Book title	Order number
Debug Tool for VSE/ESA Fact Sheet	GC26-8925
Debug Tool for VSE/ESA Installation and Customization Guide	SC26-8798
Debug Tool for VSE/ESA User’s Guide and Reference	SC26-8797
Diagnosis Guide	SC26-8528
General Information	GC26-8068
Installation and Customization Guide	SC26-8071
Language Reference	SC26-8073
Licensed Program Specifications	GC26-8069
Migration Guide	GC26-8070
Migrating VSE Applications To Advanced COBOL	GC26-8349
Programming Guide	SC26-8072

DB2 Server for VSE

Book title	Order number
Application Programming	SC09-2393
Database Administration	GC09-2389
Installation	GC09-2391
Interactive SQL Guide and Reference	SC09-2410
Operation	SC09-2401
Overview	GC08-2386
System Administration	GC09-2406

DL/I VSE

Book title	Order number
Application and Database Design	SH24-5022
Application Programming: CALL and RQDLI Interface	SH12-5411
Application Programming: High-Level Programming Interface	SH24-5009
Database Administration	SH24-5011
Diagnostic Guide	SH24-5002
General Information	GH20-1246
Guide for New Users	SH24-5001
Interactive Resource Definition and Utilities	SH24-5029
Library Guide and Master Index	GH24-5008
Licensed Program Specifications	GH24-5031
Low-level Code and Continuity Check Feature	SH20-9046
Library Guide and Master Index	GH24-5008
Messages and Codes	SH12-5414
Recovery and Restart Guide	SH24-5030
Reference Summary: CALL Program Interface	SX24-5103
Reference Summary: System Programming	SX24-5104
Reference Summary: HLPI Interface	SX24-5120
Release Guide	SC33-6211

PL/I for VSE/ESA (PL/I VSE)

Book title	Order number
Compile Time Messages and Codes	SC26-8059
Debug Tool For VSE/ESA User's Guide and Reference	SC26-8797
Diagnosis Guide	SC26-8058
Installation and Customization Guide	SC26-8057
Language Reference	SC26-8054
Licensed Program Specifications	GC26-8055
Migration Guide	SC26-8056
Programming Guide	SC26-8053
Reference Summary	SX26-3836

Screen Definition Facility II (SDF II)

Book title	Order number
VSE Administrator's Guide	SH12-6311
VSE General Introduction	SH12-6315
VSE Primer for CICS/BMS Programs	SH12-6313
VSE Run-Time Services	SH12-6312

Notices

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

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

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

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

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

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

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

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

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

Trademarks and service marks

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

CICS
CICS/VSE
ESA/390
IBM
OS/390
System/390
VSE/ESA

Index

Sending your comments to IBM

CICS® Transaction Server for VSE/ESA™

Application Migration Aid Guide

SC33-1943-03

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 anything you regard as a specific error or omission, and on the the accuracy, clarity, 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 ask questions, make comments about the functions of IBM products or systems, or to request additional publications, contact 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, to this address:

  User Technologies Department
  Mail Point 095
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER
  Hampshire
  SO21 2JN.
  United Kingdom

- By fax:
  - From outside the U.K., after your international access code use 44 1962 816151
  - From within the U.K., use 01962 816151
- Electronically, use the appropriate network ID:
  - IBMLink™: HURSLEY(IDRCF)
  - Email: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:

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

Program Number: 5648-054

SC33-1943-03