CICS® Transaction Server for VSE/ESA™

Sample Applications Guide

Release 1

SC33-1713-00

CICS® Transaction Server for VSE/ESA™

Sample Applications Guide

Release 1

SC33-1713-00

Note!

Before using this information and the product it supports, be sure to read the general information under “Notices” on page 163.

First Edition (June 1999)

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.

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.

© **Copyright International Business Machines Corporation 1989,1999. All rights reserved.**
US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Preface . . . vii
Notes on terminology . . . viii

Chapter 1. Introduction . . . 1
The CUA text level application . . . 1
The FILEA sample application programs . . . 1
The CICS intercommunication sample application programs . . . 2

Part 1. The common user access interface (CUA) sample . . . 3

Chapter 2. Introduction to the CUA guidelines . . . 5
Systems Application Architecture . . . 5
The CUA interface . . . 6
Basic mapping support . . . 8
The advantages of CUA . . . 9
Designing the user interface . . . 9

Chapter 3. BMS and CUA panel displays . . . 11
Panel elements . . . 11
Color and emphasis . . . 13

Chapter 4. BMS and CUA panel entry and selection . . . 15
Moving the selection cursor . . . 15
Selection fields . . . 16
Entry fields . . . 17
Action lists . . . 18

Chapter 5. BMS and CUA user dialogs . . . 19
Prompt . . . 19
Action bar and pull-downs . . . 19
Pull-downs and pop-ups . . . 22
Message area . . . 23
Command area . . . 24
Function key area . . . 25
Scrolling panel areas . . . 25

Chapter 6. BMS application design for the CUA entry model . . . 27

Chapter 7. BMS application design for the CUA text model . . . 29
Some application design considerations . . . 29
The end-user's view . . . 33
The designer's view . . . 47

Chapter 8. Installing and running the CUA text model application . . . 55
Generating the BMS maps . . . 55
Translating, compiling, and link-editing the application programs . . . 55
Creating the VSAM files . . . 56
Installing and running the application on your CICS region . . . 65

© Copyright IBM Corp. 1989,1999

Chapter 9. CUA text model program descriptions 67
Program DFH0VT1 – primary panel 67
Program DFH0VLST – list panel handler 69
Program DFH0VNEW – new customer panel processing 72
Program DFH0VBRW – browse customer details panel processing 75
Program DFH0VUPD – update customer record panel processing 77
Program DFH0VDEL – delete customer details panel processing 80
Program DFH0VOL – overlay handler 84
Program DFH0VOPN – open file pop-up handler 86
Program DFH0VPRT – print pop-up handler 87
Program DFH0VSAS – save customer details pop-up handler 88
Program DFH0VHLP – help pop-up handler 89
Program DFH0VHP – contextual help pop-up handler 90
Program DFH0VABT – about pop-up handler 91
Program DFH0VTBL – table router 92
Program DFH0VAB – abend handler 93
Program DFH0VRIO – customer data file handler 94
Program DFH0VLIO – help file handler 96

Chapter 10. CUA text model file and copybook descriptions 97
File: DFH0FUS customer detail file 97
File: DFH0FAI customer detail file - alternate index 97
File: DFH0FHLP help pop-up data file 97
Copybook: DFH0BCR customer record layout 97
Copybook: DFH0BCA commarea 97
Copybook: DFH0BFKT variable function key layout 98
Copybook: DFH0BFPD redefinition of file pull-down DSECT 98
Copybook: DFH0BHPD redefinition of help pull-down DSECT 98
Copybook: DFH0BHP redefinition of help pop-up 98
Copybook: DFH0BHT help file key table 98
Copybook: DFH0BLST redefinition of the list base panel 98
Copybook: DFH0BMSG application message table 98
Copybook: DFH0BRT program routing control table 99
Copybook: DFH0BTSQ TS queue details 99
Copybook: DFH0BHR help text TS queue layout 99

Chapter 11. CUA text model BMS maps 101
Map T1: map set DFH0T1 (primary panel to sample application) 102
Map LST: map set DFH0LST (list processing - base panel) 103
Map NEW: map set DFH0NEW (new customer record - base panel) 105
Map BRW: map set DFH0BRW (browse customer details - base panel) . . . 106
Map UPD: map set DFH0UPD (update customer details - base panel) 107
Map DEL: map set DFH0DEL (Delete a customer record - base panel) . . . 108
Map FPD: map set DFH0FPD (file pull-down) 110
Map HPD: map set DFH0HPD (help pull-down) 111
Map OPN: map set DFH0OPN (file-open pop-up) 112
Map PRT: map set DFH0PRT (print pop-up) 113
Map SAS: map set DFH0SAS (save changed customer record pop-up) . . . 114
Map HPOP: map set DFH0HP (contextual help pop-up) 115
Map ABT: map set DFH0ABT (about the sample application pop-up) 116
Map HLP: map set DFH0HLP (the help stub full screen pop-up) 117
Map AB: map set DFH0AB (abend handling) 118

Part 2. FILEA sample applications 119

Chapter 12. Installing and running the FILEA sample applications . . . 121
Installing the sample groups 122
Language considerations 122
Running the sample applications 123

Chapter 13. FILEA sample application program descriptions 125

Chapter 14. FILEA sample application file description 131

Chapter 15. FILEA sample application BMS maps 133

Part 3. Intercommunication sample applications 135

Chapter 16. The intercommunication sample applications 137
Intercommunication sample 1 – temporary storage queue transfer 138
Intercommunication sample 2 – remote file browse 140
Intercommunication sample 3 – remote record retrieval 142
Intercommunication sample 4 – CICS to CICS conversation 144

Part 4. BMS partition and transient data samples 151

Chapter 17. The BMS partition samples 153

Chapter 18. The transient data sample (DFH$TDWT) 155

Bibliography 157
Books from VSE/ESA 2.4 base program libraries 158
Books from VSE/ESA 2.4 optional program libraries 160

Notices 163
Trademarks and service marks 164

Index 165

Preface

What this book is about

This book is about the CICS sample application programs that are supplied to assist you with application program development.

Who this book is for

This book is for those responsible for designing and writing CICS applications programs and for those responsible for installing and running the supplied sample applications.

What you need to know to understand this book

The book assumes that you are a CICS application programmer. You should be familiar with the CICS application programming interface (API).

How to use this book

Each part of this book describes a separate sample. Read whichever part is relevant to your current task.

© Copyright IBM Corp. 1989,1999

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

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 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	
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-conforming version of the C programming language compiler. Generally referred to as “C/VSE”.
IBM COBOL for VSE/ESA	The Language Environment-conforming version of the COBOL programming language compiler. Generally referred to as “COBOL/VSE”.
IBM PL/I for VSE/ESA	The Language Environment-conforming 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, generally referred to as “SAF”
VSE/ESA Central Functions component	The new name for the VSE Advanced Function (AF) component
VSE/VTAM	“VTAM”

Chapter 1. Introduction

This book describes the sample applications supplied with CICS Transaction Server for VSE/ESA Release 1. It is presented in three parts, each dealing with a different type of application. These are:

1. A complete example of a CUA® text level application
2. Four sets of command-level application programs that operate on the sample VSAM file FILEA
3. A set of command-level application programs illustrating the use of CICS® intercommunication facilities.

The CUA text level application

CICS provides a sample application to demonstrate BMS support for the Common User Access® (CUA) interface. The application uses an action bar, with associated pull-downs, pop-ups, and help panels. The application programs demonstrate how to code COBOL/VSE programs to display, overlay, and remove CUA style windows.

The main objective is to show what can be implemented, and the amount of design effort that is involved. You can take these applications and enhance them to meet your own requirements by referring to the *CUA Basic Interface Design Guide*. You can also use the samples to evaluate BMS support for the CUA interface.

The basic characteristics of the application are:

- It is written in COBOL/VSE
- It is pseudo-conversational.
- It uses minimum function BMS.
- It uses VSAM KSDS files.
- It uses communication areas and TS queues to maintain the status and position of the user in the dialog.

The FILEA sample application programs

CICS provides four sets of command-level application programs that operate on the sample VSAM file FILEA. There is one set for each of the four programming languages supported, (Assembler, C, COBOL, and PL/I). Each set comprises the following six programs:

- Operator instruction
- Inquiry/update
- Browse
- Order entry
- Order entry queue print
- Low balance report.

These programs show basic functions, such as inquire, browse, add, and update, that can serve as a framework for your installation's first programs. They were all written prior to the publication of the Common User Access guidelines. This application can also be used as an Installation Verification Procedure.

© Copyright IBM Corp. 1989,1999

The VSAM file, FILEA, consists of records containing details of individual customer accounts.

The CICS intercommunication sample application programs

CICS provides four intercommunication sample application programs, written in assembler language, that illustrate the use of distributed transaction processing and asynchronous processing on APPC and LUTYPE6.1 links.

The four applications demonstrate the following functions:

1. Transferring a temporary storage queue from a local CICS system to a remote CICS system, using distributed transaction processing and APPC protocols.
2. Browsing a remote file, using distributed transaction processing and APPC protocols.
3. Retrieving a record from a remote temporary storage queue, using asynchronous processing. This sample can be used with APPC and LUTYPE6.1 links.
4. Enabling a CICS-to-remote LUTYPE6.1 system conversation. LUTYPE6.1 links must be used for this sample.

Part 1. The common user access interface (CUA) sample

This part of the book shows how you can design CUA conforming application programs that communicate with nonprogrammable terminals using CICS basic mapping support (BMS).

The CUA interface is fully described in:

- *CUA Basic Interface Design Guide*, SC26-4583.
- *CUA Advanced Interface Design Reference*, SC34-4290.

The CICS-supplied CUA text model application discussed in this book is based on that part of the CUA guidelines that is applicable only to nonprogrammable terminals, as described in the *CUA Basic Interface Design Guide*.

There are several solutions now available to the application programmer to enable CICS programs to display information to the end user, using methods other than BMS. The main objectives here, are to provide guidance on:

- Writing applications programs that conform to those parts of the CUA guidelines that can be implemented in the CICS environment using BMS
- Designing applications in a modular fashion, with front-end and back-end components to manage terminal and disk I/O respectively
- Designing applications that are portable between different CICS platforms, and can exploit CICS multiregion operation (MRO) and ISC facilities.

This part of the book contains the following chapters:

- Chapter 2, “Introduction to the CUA guidelines” on page 5
- Chapter 3, “BMS and CUA panel displays” on page 11
- Chapter 4, “BMS and CUA panel entry and selection” on page 15
- Chapter 5, “BMS and CUA user dialogs” on page 19
- Chapter 6, “BMS application design for the CUA entry model” on page 27
- Chapter 7, “BMS application design for the CUA text model” on page 29
- Chapter 8, “Installing and running the CUA text model application” on page 55
- Chapter 9, “CUA text model program descriptions” on page 67
- Chapter 10, “CUA text model file and copybook descriptions” on page 97
- Chapter 11, “CUA text model BMS maps” on page 101.

The book attempts to avoid duplicating any information that is already contained in the *CUA Basic Interface Design Guide*. Any CUA information that is repeated here is for the purpose of making specific comments, or where recommendations are made.

As the CUA rules and definitions are defined in the *CUA Basic Interface Design Guide*, this book confines itself to offering the application designer guidance about the practical techniques that can be used with BMS support, or within the application program, to create consistent CUA applications.

© Copyright IBM Corp. 1989,1999

Chapter 2. Introduction to the CUA guidelines

It is commonplace today to find users running programs on workstations, such as personal computers (PCs), that exploit windowing. These take full advantage of high performance graphical displays and their associated interface techniques. When a PC user switches from a PC-based application program to a host-based program, it is important that consistent user interface techniques apply across the two different environments—PC and host. For example, the common use of function keys and color, and panel layout standards, help the user to move easily from one environment to the other. Part 1 of this book is intended to help you to design applications to a consistent CUA standard at the CUA entry level[1], and applications at the text level[1] that can contain action bars and associated pull-down and pop-up windows. The CICS supplied CUA text model application discussed in this book illustrates the common dialog flows that these CUA techniques entail.

To change an existing application to CUA standards means making a number of adjustments to the application. For example, using an application action bar (AAB) provides a *fast path* to many places in the application. Because this cannot be done locally on a non-programmable terminal, the work must be done by the CICS application program. In the text level environment, instead of BMS sending and receiving a few modified fields to the same map, it may now have to send different maps to manage the required screen presentation.

It must be stressed that although BMS has little built-in support for the CUA interface, there is enough function in BMS to allow you to include all of the CUA panel elements in your BMS maps. You can migrate your CICS applications to CUA standards by using normal BMS coding practices. However, it is unlikely that you can achieve all of the possible CUA requirements as long as the application has to run on 3270-type terminals, because some of the more advanced CUA functions are only suited to the personal computer with an all-points-addressable screen and its capability of giving rapid locally-controlled interaction with the end user.

You must evaluate the benefits of an improved end user interface against the possible increases in processing time, line traffic, and development time for the application code. When assessing the question of performance, you must take into account not only specific transaction counts, but also the cost in terms of time and money of retraining operators on applications that all work in different ways.

Systems Application Architecture

The Systems Application Architecture was announced by IBM® in March 1987 and CICS is a product conforming to these rules. The SAA architecture is a set of software interfaces, conventions and protocols with an initial goal of providing a framework for designing, developing, and using (or interacting with) applications with cross-system consistency. This book is concerned only with the common user access portion of SAA, and using the CUA interface in the CICS environment.

---

1 Entry level and text level are specific models defined in the CUA interface. For information about these CUA terms, see the *CUA Advanced Interface Design Reference* and the *CUA Basic Interface Design Guide*.

© Copyright IBM Corp. 1989,1999

The CUA interface

CUA is a set of rules, guidelines, and options that, if adhered to, enable you to provide applications with a consistent appearance to your end-users.

CUA distinguishes between programmable workstations, such as PCs, and nonprogrammable workstations, such as host-attached 3270-type terminals. A PC connected to a host using 3270 emulator programs is regarded as a nonprogrammable terminal by an application such as CICS.

The *CUA Advanced Interface Design Reference* provides guidance for application programmers who wish to incorporate CUA interface design into their applications. The *CUA Basic Interface Design Guide* specifically addresses the use of CUA for nonprogrammable terminals.

Within the host environment, CUA also distinguishes between entry level and text (or *character*) level.

CUA consistency in CICS applications

For the purpose of this discussion, consistency means the CUA conventions for common operations even though the underlying techniques might vary. For example, there are several ways of saving data: on various types of local file systems; on remote file servers; or on database servers. CUA guidelines state that end-users at terminals should not be aware of these technical details. In this example, the user is storing data but does not need to know how or where.

There are three main aspects of CUA consistency:

1. Physical
2. Syntactical
3. Semantical.

Physical consistency applies to hardware; for example, keyboards having function keys consistently labeled and positioned.

Syntactical consistency applies to the order and appearance of fields on the screen, such as placing the title at the top, centered and correctly colored.

Semantical consistency refers to consistency in the use of words, such as the meaning and use of *cancel* or *exit* commands.

Guidelines are published and examples provided for the application designer. However, it is easy to assume that your situation is different from the norm, and therefore not addressed by the guidelines. For example:

- Database applications don't *save* data, they *update* it
- Some applications don't *print*, they *plot*
- Some applications don't *delete*, they *clear*.

Sometimes the distinctions are important because of the convention for the industry for which the application was designed. However, quite often the terminology has historical significance that is no longer relevant. The simplicity and consistency of the CUA interface should appeal to new users who don't know or don't care about the historical significance of CICS application terminology. On the other hand, the

terminology cannot be ignored altogether because non-CUA users should be able to switch to a CUA-based application without having to relearn everything.

The discussion that follows considers two categories of users:

1. Those who know about the CUA interface and:
   - Know about CUA applications and their concepts
   - Have no previous knowledge of CICS applications and their concepts.
2. Those who know about CICS and:
   - Know about CICS applications and their concepts
   - Have no previous knowledge of CUA applications and their concepts.

You may need to take these categories of user into account when redesigning CICS applications. Is it a CICS application that has to be adapted to CUA, or a CUA application that now has to use CICS? If it is the former, you tend to use primarily CICS terminology, but adopt CUA terminology when it is convenient (in effect, making only a token gesture to the use of CUA in your applications). If it is the latter, you tend to keep to the CUA user's view of the interface, and present the CICS operations in a CUA manner.

If you emphasize CICS terminology:

- Users with previous CICS experience, but no CUA experience, have a better chance of understanding the functions of the application.

  If the terminal user exclusively uses CICS applications, the fact that it operates under CUA guidelines is unimportant. This is a limited perspective, however, for although it is true that some users may begin in this way, an objective of CUA is to encourage the wide use of other applications. A CICS application objective should be to have its operational model consistent with other CUA applications.
- Users with CUA experience but no CICS experience could be put off by CICS-based terminology. Their ability to apply previous CUA experience is limited, as is their ability to use the application, because they must think about the actions (for example, normal pull-down interactions) that are automatic in other applications.

The CICS application biased in this way does little to prepare users for other CUA applications.

On the other hand, if you emphasize CUA terminology:

- Users with previous CICS experience and no CUA experience might not recognize a once-familiar application.
- Users with CUA experience and no CICS experience will feel more familiar with the application because of their ability to recognize application actions from previous CUA applications.

The CICS application biased in this way helps develop an expertise that can be applied to other applications.

You should aim at developing applications that are biased towards the user who is familiar with CUA because:

- The number of users who are accustomed to non-CICS CUA applications is growing.
- The majority of applications will be consistent with one another if you follow the guidelines.
- Cross-application consistency encourages users to try other applications.
- In time, the most successful applications will evolve to CUA standards and these standards will continue to be adjusted to real world usage.
- Programming techniques developed for one CUA application can easily be propagated to further CUA applications.

Recommendation

When designing a CICS CUA text-level application, use the standard actions and pull-downs wherever CUA concepts are applicable.

Provide the CUA standard actions in the pull-downs whenever possible, and augment the titles when necessary. For example, if it is necessary for a user to declare an intent, the intent should be included in a pull-down using phrases such as “Open for Update” or “Open for Browse.”

Functions that are unique to the application but represent activities that are in the same categories as the standard pull-downs, should be added in the relevant places.

Basic mapping support

Basic mapping support (BMS) is the CICS interface that formats input and output display data between CICS and an application program. It supports minimum function character-mode non-programmable terminals, although BMS applications can also run on PWSs under CICS OS/2™.

BMS exists in three pregenerated versions: minimum, standard, and full. The version used in a CICS region is determined by the BMS system initialization parameter. Each version provides a different level of function, and consequently uses different amounts of virtual storage, the minimum level of BMS using considerably less than the other two levels.

In general, the guidance in this book covers what you can do using minimum BMS, which supports the following:

- The SEND MAP command
- The RECEIVE MAP command
- The SEND CONTROL command
- Default and alternate screens
- Extended attributes
- Map set suffixes
- Field and block data.

Minimum-level BMS supports all 3270-type displays except SNA character string printers. There is little CUA advantage to be gained from any of the additional facilities offered by the standard and full versions of BMS.

The 3270 terminal devices (or 3270 PC emulators) without *extended attribute* support do not conform to CUA. On the other hand, 3270 devices support some functions such as partitions and partition scroll keys, validation attributes, and light pen/cursor select fields, that are not part of CUA.

For information about BMS see the *CICS Transaction Server for VSE/ESA Application Programming Guide*.

The advantages of CUA

A consistent interface benefits users and application designers, saving both time and money.

Users benefit because they need less time to learn how to use an application and, when using the application, take less time doing their work. An additional benefit to users is reflected in their attitudes. A consistent interface reduces users' frustration levels, increases their feeling of accomplishment, and makes them feel more comfortable with the system.

Application designers benefit because a common building-block approach can be defined for an interface using standardized interface elements and interaction techniques. The building blocks allow programmers to create and change applications more easily and quickly. The same designs and techniques used across many systems enable application designers to reuse BMS elements and modules of associated application code.

Designing the user interface

The major parts of CUA interface design are panel design, entry and selection design, and dialog design.

To give some idea of what is involved in using CUA using BMS, the different levels of CUA implementation complexity are discussed here under four *thresholds*.

Threshold 1: This defines the straightforward changes that can be achieved by re-designing the BMS maps and making minor changes to the dialog flow of the program. This level is recommended for converting existing CICS applications because it conforms to the CUA entry-level model and gives significant benefits to the conventional data-entry type of operation.

Thresholds 2 and 3: These are more complex and suitable only for new or significantly re-designed applications. These thresholds conform to the CUA text-level model and give significant benefits to terminal users. For these, a knowledge of *object-action* and *action bar* design is necessary.

Threshold 4: This lists CUA elements that are ***not*** recommended because the effort will not justify the results.

The following is a summary of the CUA content of these thresholds, (where each threshold includes the function of the lower threshold):

Threshold 1 (entry level — existing applications): Defines changes that are straightforward to implement by re-designing BMS maps, and by making minor changes to the dialog flow of the program. This involves:

- Using the required basic panel elements and formats
- Setting the required colors on initial presentation
- Handling simple list panels and elementary scrolling
- Moving any command and message areas to the correct places
- Changing the current function keys to use the required dialog definitions and show them in the correct area
- Using full screen help.

Threshold 2 (text level): Defines changes that are more complex to implement. There is an increase in the number of maps, and a corresponding increase in the program flow path. This involves:

- Changing to an object-action approach.
- Incorporating action bars and their associated pull-downs.
- Maintaining underlying color consistency when a pull-down or pop-up is present, and reflecting the correct selected emphasis.
- Using full dialog control. This includes protecting the user against the consequences of using all inoperative function keys (for example, the “Clear” and “PA” keys).

Threshold 3: Defines changes that are possible but complex to implement. It involves:

- Incorporating pop-up windows
- Handling list panels with multiple actions against listed objects
- Implementing pop-up help panels on a field-context basis.

Threshold 4: Defines CUA items that are either impossible to implement, (because of CICS, BMS, or 3270 limitations) or the effort involved does not justify the potential benefits. It involves:

- User-customized options; for example, setting the display panel id on or off, or using different displays of the function key line.

The CUA-preferred method is for an object-action approach as opposed to the action-object method. The object-action method allows users first to select an object from the panel body, and then to select an appropriate action from the action bar or function key area to work on that object. This generally results in fundamental application design changes to the panel layouts and the underlying dialog control. The CUA text model application program described in this book uses the object-action method.

Chapter 3. BMS and CUA panel displays

This chapter covers the various CUA-defined panel elements that may be used in BMS maps, and also the use of color and emphasis.

Panel elements

The following panel elements, the rules for which are defined in the *CUA Basic Interface Design Guide*, are covered here:

- Panel identifier (or id)
- Panel title
- Panel area separators
- Instructions
- Column headings and group headings
- Field prompts
- Descriptive text
- Protected text.

You can define all of these using BMS.

Panel identifier

BMS enables you to define fields that comply with the CUA rules for panel identifiers. However, if you want the terminal user to select whether or not the panel id is displayed, you must code the application program to test an appropriate indicator, and change the 3270 attribute to hide the contents of the field accordingly.

Recommendation

Code a panel id that is permanently displayed in a panel. If you want the users to be able to turn the panel id off, they should have the option when they enter an application for the first time, and their choice should then remain set for the duration of the session.

Choose a panel id that relates to the transaction id, and which is therefore meaningful to the application program. This assists the control of dialog flow within the application.

Panel title

BMS enables you to define fields that comply with the CUA rules for panel titles. It is your responsibility to position the panel title field in the center of the panel. For this purpose, you might find that a utility such as the IBM Screen Definition Facility II (SDF II) makes screen definition (or *painting*) of your CUA/BMS maps easier than using the CICS DFHMDF macros since SDF II supports centering.

Because variable window sizing is not possible using BMS maps, you do not have to adjust the position of the title when a window size changes. For further information see “Scrolling panel areas” on page 25.

You must ensure that if a map contains a variable length insert the application program centers the insert as nearly as possible.

© Copyright IBM Corp. 1989,1999

Panel area separators

The CUA interface requires you to separate the action bars from the body of the panel.

Recommendation

Use a line of hyphen (-) symbols as the separator between the action bar and the main panel body, but in other places on the panel use blank lines. Define the separator symbol line as a protected field, and part of the map definition.

Instructions

BMS enables you to define fields that comply with the CUA rules for instructions. Each line of instruction text is preceded by an attribute byte. BMS does not allow fields to extend beyond the end of a line.

Recommendation

Define areas of instruction text as protected fields and part of the map definition.

Column headings and group headings

BMS enables you to define fields that comply with the CUA rules for column and group headings.

Recommendation

Define column and group headings as protected fields and part of the map definition.

Field prompts

BMS enables you to define fields that comply with the CUA rules for field prompts.

Recommendation

Define the field prompts as protected fields and part of the map definition. You should also define any leading dots as part of the field prompt.

Descriptive text

BMS enables you to define fields that comply with the CUA rules for descriptive text.

Recommendation

Define areas of descriptive text as protected fields and part of the map definition.

Protected text

BMS enables you to define fields that comply with the CUA rules for protected text.

Recommendation

Define areas of protected text as protected fields and part of the map definition.

Color and emphasis

The color and emphasis information in the *CUA Basic Interface Design Guide* describes the default colors and emphasis techniques assigned to the panel areas and elements. As users progress through a dialog with the application, the colors and emphasis may change to show the current status of an element. As an application programmer you are responsible for ensuring compliance with the color and emphasis standards described in the CUA manuals.

It is usually possible to use a single BMS map, within a particular application program, that supports both the monochrome and color environments. For example:

- In CUA, unavailable choices in a selection list are colored blue for color devices, or are indicated by placing an asterisk over the first character for monochrome devices.

If you develop an application that is designed for 3270-type color devices, BMS will filter out any unsupported attributes before transmitting the data stream to the device. This means that panels will display on all devices, but may not fully comply with CUA for monochrome devices (for example, the use of underscore attributes for entry-field sizes have no effect on a 3277 terminal). However, you can specify high intensity in conjunction with color attributes, so that the map is generally suitable for use on both monochrome and color 3270-type terminals. When using a 7-color 3270-type terminal, the high-intensity attribute is ignored. Nevertheless, you should use high intensity for all messages.

Recommendation

For BMS applications, you should assume that the target device supports color, and set the color attribute as appropriate. However, in addition to setting color attributes, you should also use the monochrome technique of displaying an asterisk to indicate which choices are unavailable. Otherwise, the user of a monochrome device will not be able to detect that a choice is invalid unless he selects it and causes an error message to be returned.

Because the CICS supplied CUA text model application uses the recommended CUA color emphasis ***and*** overwrites selection numbers with an asterisk, it is suitable for both color and monochrome devices.

Chapter 4. BMS and CUA panel entry and selection

The *CUA Basic Interface Design Guide*. explains selection fields and entry fields. On a non-programmable terminal the cursor character generated by the terminal hardware performs a dual function: (1) pointing to objects, and (2) indicating the user's current input position. Using the cursor keys, the cursor can be moved anywhere on the screen, and not just to entry and selection fields. This adds to the complexity of interpreting exactly the user's action when a CUA-conforming BMS map is returned to the application.

Moving the selection cursor

There are several situations where the position of the cursor on input is significant. These include:

- Selecting an action from the action bar.
- Entering a choice in a selection field, when there is only one selection field in the panel.
- Requesting contextual help. If the user presses the help key (F1) when the cursor is within an entry field, the application should provide help that is specific to the field. If the cursor is not within an input field then the application should provide general, panel-related help.
- Selecting a function key by placing the cursor in the function key area and pressing enter.

The cursor position is passed to the host as part of the inbound 3270 data stream, and for CICS applications (using terminal control or BMS) it is available:

- As a fixed binary halfword value in the field EIBCPOSN. If your application knows the format of the panel, it can use the EIBCPOSN value to determine in which field the cursor was positioned. However, if the screen format is changed at any time, the application program logic which identifies the cursor position must also be changed. This technique means that programs have to be changed even though the application data structure for a map does not change.
- In a flag byte of the field in which the cursor was positioned. To use the flag byte for this purpose, you must specify the cursor location option (CURSLOC=YES), in the BMS map definition, using either the DFHMSD macro, the DFHMDI macro, or both, depending on your requirements.

  (See the CICS Transaction Server for VSE/ESA Application Programming Guide for more information about the flag byte which BMS uses to indicate to your application the field in which the cursor was located.)

Detecting the cursor on action bars: Under CUA rules, the names for action bar options are largely predetermined, and should have a single blank character before and after each. However, if your maps are translated into different languages, the lengths of action names are almost certain to be different for each language. If the lengths of the action names in your maps change as a result of translation, their position on the screen will also change, and if you are using the EIBCPOSN value to detect the selected action bar choice, you must change your application program

© Copyright IBM Corp. 1989,1999

accordingly. This is not the case if you are using the CURSLOC option, which provides program independence from such map changes.

Testing for the cursor position

A MAPFAIL condition results in a zero value for the cursor position in EIBCPOSN. This can be caused by, for example, repeated use of the CLEAR key, and your application program should test for the possible use of the CLEAR key before making tests on the cursor value.

If

- You are using the cursor flag option, and
- The cursor is positioned in one field of the map, and
- None of the fields contains data

BMS sets the cursor flag for the appropriate field and all the fields in the application data structure are set to null. In this case, the MAPFAIL condition is ***not*** raised. The unmapped data stream is not available to your application program unless it issues an EXEC CICS RECEIVE request.

Other selection considerations

The 3270 architecture also provides a means of selecting a field using the cursor-select key (CURSOR SEL), which provides the same function as the light pen. BMS supports the use of the cursor-select key, and if you want to use this method of field selection, you must ensure that the field is defined as detectable when you specify the BMS map. (See the CICS Transaction Server for VSE/ESA Application Programming Guide for more information about defining fields to use the cursor-select key.)

In the case of a non-programmable terminal you must define the fields that represent the available choices in an auto-select field as unprotected, to allow use of the tab keys to skip from one choice to another. This allows the user to type over the predefined choices, even though the application does not expect, and should ignore, any data entered by such overtyping. If the user overtypes selection data, and the panel remains displayed, the original choice data will not (necessarily) be redisplayed.

Selection fields

CICS does not preclude the use of CUA facilities such as icons, checkboxes, and radio buttons (provided that your map is able to return, and the application can correctly interpret, the correct cursor position). Some 3270-type terminals have the ability to display graphics, and make these CUA techniques feasible. However, the practical considerations and cost of sending complex graphics data over a typical telecommunication network rule these out. For this reason, the CICS sample applications support only character mode, and the selection choices are straightforward text.

The sample applications use entry fields for selecting choices from a list. In a list offering a single choice, there is one entry field, to the left of the first choice in the list, in which the user can enter the selection number. In a list offering multiple choices, there is a one-character entry field to the left of each available choice, in which the user can enter a selection character.

The use of capital letters is a matter for individual choice.

Scrollable selection fields and lists

Scrollable selection fields and lists can be implemented under BMS, but you must balance the cost, in terms of application code and performance overhead, against the potential value to be gained in terms of usability.

Selection element emphasis

The selection cursor of a non-programmable terminal is the text cursor, and you cannot control its appearance by your application code.

Your application must ensure that the selection characters are redisplayed with the correct emphasis if the panel remains in view after the user has selected a choice.

Unavailable emphasis is a de-emphasized color, for color character devices, and an asterisk overlaying the first character of the choice text, for monochrome character devices. Your applications need to be sensitive to the device type to determine how to mark invalid choices. If a user makes an invalid choice, your application should display a message either on a predefined message line or in a pop-up. This means that you should give a field name in your BMS map to any selection fields that are potentially not available.

Selection field initial conditions

There are no specific CUA rules about setting initial or default values for selection fields, so you can choose whether or not to do so.

Entry fields

In CUA-based BMS applications you will frequently send maps without the ERASE option on the SEND command. This means that any entry fields in panels that you send to a terminal, must be filled with blanks to ensure that they correctly overlay any existing maps, and do not leave old data displayed unintentionally. When receiving data from a map, your application program should treat a blank in an entry field as the equivalent of a null field.

Entry field appearance

You should use underscore attributes in BMS maps to indicate the extent of entry fields. This requires terminals that support extended-highlighting attributes (terminals without extended-attribute support do not conform to CUA requirements). If a terminal does not support extended highlighting, you can obtain the effect of the underscore attribute by filling entry fields with underscore characters. However, this may not be acceptable because insert mode is then not readily available.

You can implement scrollable entry fields under BMS but this requires considerable effort in programming, and increased overhead in terms of performance. Therefore the use of scrollable entry fields is not recommended.

Action lists

BMS allows the definition of fields that comply with the rules for action lists, but you must write the code to support them.

Chapter 5. BMS and CUA user dialogs

This chapter discusses how you handle CUA user dialogs using BMS. The following topics are discussed:

- Using prompts
- Using action bars and pull-downs
- Using pull-downs and pop-ups
- Using message areas
- Using command areas
- Using function keys
- The use of scrolling.

Prompt

The *CUA Basic Interface Design Guide* discusses the use of prompts in CUA dialogs to assist terminal users in completing entry fields. Prompts can save time for users and reduce the risk of typing errors. CUA specifies that anything other than very short prompts (which can usually appear as descriptive text) should appear in pop-up windows.

There are no BMS restrictions in implementing CUA prompts, because all the code to handle prompts must be in the application program.

Action bar and pull-downs

The *CUA Basic Interface Design Guide* discusses the requirements for an action bar. Although simple applications can be developed using the CUA entry level model (that is, without an action bar), to implement a CUA text level model, an action bar must be used.

Use of the action bar and pull-downs

A terminal user can select an action from an action bar by tabbing to the action field and pressing ENTER, or by setting the cursor anywhere within the action name and pressing ENTER.

Using BMS, your application can either:

- Receive the cursor position as a halfword binary value, in the field EIBCPOSN, or
- Receive the cursor position using the cursor location option.

The application then has to determine which action the user selected.

BMS action bar fields must be defined as unprotected, to enable the user to select an action by means of the tab key. However, this means that the user could overtype, and even completely erase, the action bar options. The application must be capable of restoring these correctly on the next interaction. Also, using action bars with associated pull-down windows adds complexity to the application code and increases the inbound and outbound data flow to and from the host.

© Copyright IBM Corp. 1989,1999

Action bar layout

An example of a standard action bar layout is shown in Figure 1

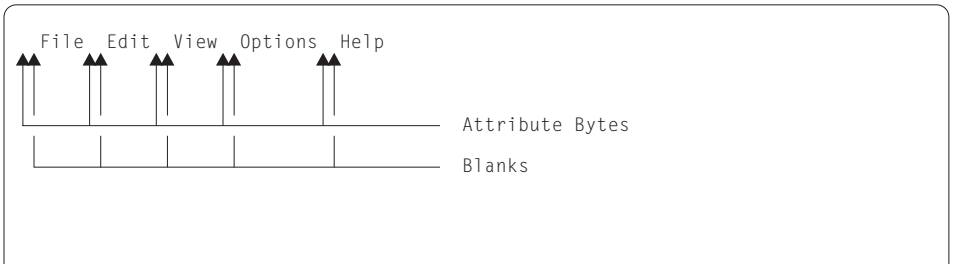


Figure 1. An example of a standard action bar

The CUA requirements for action bar attribute handling are supported by BMS. CUA demands a solid bar separator for various purposes, and solid lines and corners for the borders of pull-down windows. Because neither the minus (–) nor the underscore (_) symbols produce a solid line when repeated, and because there are no horizontal or vertical corner-frame characters available that display on all 3270 devices, you must decide for yourself what gives the best presentation to the user, within the CUA guidelines.

Recommendation

If you know that your application is only available on a particular set of terminals that support a character set with proper box edges, corners, and intersections, you can code the application to use the character set as defined by CUA. However, if the application has to support a mixture of terminals, some of which have the correct character set and some which don't, use the second of the CUA-preferred borders for your pull-downs; that is:

- Use the minus (–) for horizontal lines
- Use the vertical bar (|) for vertical lines
- Use the period (.) for the top corners
- Use the apostrophe (') for the bottom corners.

Action bar content

CUA recommends that you should use the standard actions if at all possible. These are:

File Edit View Options Help

When planning an action bar for a CICS application, you may be tempted to keep to terminology with which the users are familiar. However, you should make every effort to adopt the standard CUA action bar, because that is what the users are most likely to come across in other CUA applications. If your application does not need to use a particular action, omit that action's name from the action bar and space out the remaining action names so as to leave only the attribute byte and a single blank between each.

Action bar selected emphasis

If you design an application that supports pull-downs, you must ensure that the selected action on the action bar is emphasized while its pull-down is displayed. To do this, your application must be able to set the color and extended highlighting attributes appropriately. This means you must define the action fields in the BMS map as modifiable (by the application program).

How users interact with the action bar and pull-downs

When a user selects an action from the action bar the application should display a pull-down in which further actions are presented as a list. (A common mistake is to trigger actions directly from the action bar, rather than from actions listed on the resulting pull-down.) The contents of the action bar pull-down are in the form of a list, which is usually a numbered, single-choice selection list, but can be a multiple-choice selection list.

The entry field in the pull-down must be aligned directly under the blank character which precedes the selected action's name. See Figure 2.

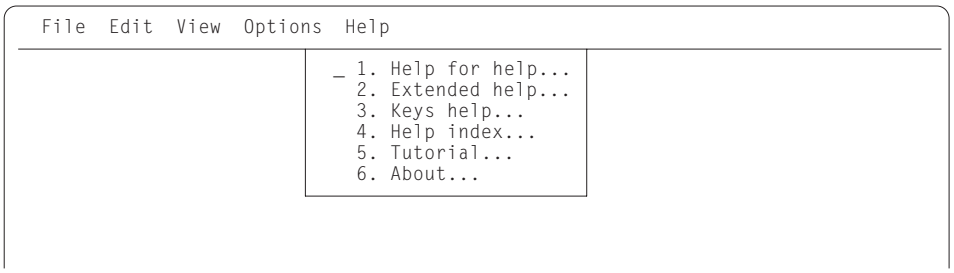


Figure 2. Example of a pull-down panel

To enable users to switch the cursor from the body of a panel to the action bar, the function key F10 (Actions) must act as a toggle switch; therefore, the application must detect the switch-to-action-bar key, save the current cursor position, and place the cursor on the first action in the action bar. If, having responded to F10 in this way, the application detects F10 again, it must restore the cursor to the saved position in the panel body. Furthermore, if the user makes a selection from the action bar that results in a pull-down, and then presses F10, the application must clear the pull-down and still restore the cursor to the saved position in the panel body (not to the action bar).

Rules for user interaction using a keyboard

The 3270 hardware determines how a user may move the cursor to the action bar. In many cases a tab (or backtab) key produces the quickest result, and therefore is the method most likely to be used. (Using F10 requires an interaction with the host.) For this reason the application should not be designed to rely only on an F10 interrupt; it must be able to determine the cursor position whenever an attention key is pressed.

Pull-downs and pop-ups

The *CUA Basic Interface Design Guide* discusses the location, layout, and content of pull-down and pop-up windows.

You should not attempt to develop a text-level application using pop-up windows without carefully analyzing the amount of effort involved, and considering the overhead incurred in terms of extra line traffic. Although BMS does not support true *windows*, it can be used to simulate windowing, and so give the application a consistent CUA appearance. However, you cannot prevent the user from moving the cursor outside the boundary of a BMS-generated pop-up window.

Neither CICS nor BMS provides a windowing environment and therefore cannot support secondary windows. BMS can be used only to provide pop-up window support if the application saves the screen contents from each transmission. Pop-up window support means supporting pop-up windows (with borders) which overlay only part of the main screen, and disable any entry fields in the main panel. This will increase path lengths and storage requirements for the application.

You could consider a limited form of pop-up for help information, where the help window overlays the main map, but your application program must still be capable of re-displaying the original map when the help pop-up is canceled.

An example of this can be seen in “Contextual help” on page 45.

Pull-down and pop-up techniques using BMS

In the discussion that follows, the term *base map* means a normal BMS full-screen map.

If you decide to use simulated windows, you can achieve this in one of two ways, which are referred to as the *canned map* and the *overlay map* techniques. Using BMS, you handle pull-down and pop-up windows in the same way.

Canned map
With this technique, you copy the base map and define the window as an integral part of it. This may give easier control, but it increases the number and complexity of the maps, and means that for any subsequent changes to the base, every version must be updated.

Overlay map
With this technique, the map consists only of the window and its contents. The map must be coded completely, with any entry or unprotected fields space-filled to prevent underlying information in the base remaining visible. For example, you must initialize single byte stop fields with a space. To display the overlay window, send the overlay map and omit the ERASE option. This method minimizes the number of maps, and eases maintenance, but it makes control of the underlying panel more complex, because the base map remains on the screen. Although the base map remains on the screen, it does not exist to the application, and any RECEIVE MAP command operates on the window map. However, you cannot prevent the user from moving the cursor to fields in the base map and attempting to enter data that is not available to your application.

To the terminal user, these two techniques produce the same effect.

Recommendation

Use the overlay technique, as demonstrated in the CICS sample CUA programs.

In normal operation there will be occasions when the user's terminal displays several maps; for example, the base panel, a pull-down, and perhaps a pop-up. If the application keeps a record of these maps, it can handle functions such as "clear" by refreshing the screen and resending the component maps.

Use temporary storage queues to keep track of the panels for re-building maps.

Message area

Usually CICS applications communicate with users by sending messages. Generally, these messages present users with unsolicited information — information that you, the application designer or programmer, believe users need to know.

Types of messages

The *CUA Basic Interface Design Guide* defines three types of messages, defined according to their severity, and what users do to remove the message. The message types are:

1. **Information**. These messages tell users that a computer function is being performed or has completed normally.
2. **Warning**. These messages tell users that a potentially unwanted situation could occur. Users do not need to correct the situation in order to continue, although they may need to take corrective action later to avoid errors.
3. **Action**. These messages tell users that an exception condition has occurred. Users must take some action to correct the situation. Action messages are used in situations ranging from minor application-related conditions (that prevent users continuing with the current dialog), to system-related conditions (that prevent users from continuing to work with any application in the system).

There is no automatic generation of the required attributes for the different categories of message. The application must set the attribute characters for the fields containing its own message lines.

The above mentioned messages are application messages. Any CICS-generated messages overwrite part of the screen, but you can control this to some extent by specifying the *error last line* diagnostic option in the terminal's typeterm definition. Because CICS messages are generally action messages (as defined by CUA) you must also set the *error color* option to red.

Message location

Messages are normally displayed in a message area in the base map rather than in pop-ups, because of the overhead associated with pop-up support. This means that scrolling the message area is less of a problem than for some of the other panel areas, because the message area will consist of one or more single field lines (that is, all message lines are formatted identically). You must ensure that the application program is able to determine when scroll keys are used, whilst the cursor is positioned in the message area, and update the contents of the message area appropriately.

Message content

Information messages should contain text only and should be displayed in message lines in the main panel.

Warning messages should contain text only, and should be accompanied by an audible “beep,” which can be achieved by specifying the *alarm* option on the BMS SEND command.

Message removal

The application must ensure that the message text is removed from the screen when appropriate, as defined by the CUA rules, and that help information and scrolling are dealt with.

Recommendation

Define all maps with a fixed message line. This includes coding pull-downs and pop-ups with a single message line on the same line as that on a base map. When a base map is displayed, the message line can be filled in the normal way. When a pull-down or pop-up is displayed over a base map, (and the base map has ceased to exist as far as the program is concerned), the message line from the pop-up or pull-down will overlay the message line of the base map.

Audio interaction

You can sound the terminal audio warning (known as the *alarm* or *beep*) by issuing an EXEC CICS SEND CONTROL ALARM.

Copyright information

You can use BMS for CUA copyright information requirements by sending a message to the applications primary map.

Command area

The *CUA Basic Interface Design Guide* discusses the requirement for command areas. These can be used in an application to allow direct entry of commands, in addition to using action bars and pull-downs.

You can decide on the location and layout of the command area, because BMS does not distinguish the command area from any other fields in the map.

If you want to support a *retrieve last command* key, this must be implemented in your application code.

Because BMS does not allow a field to “wrap around” from the end of one line to the start of the next, a two-line command area would be complicated to implement. The application would have to join the two fields together before interpreting the command, and the result of this may not be as desired; for example, delete- and insert-key functions would not apply to the whole command area.

Recommendation

Use a fixed location, single-line, command area, rather than a pop-up, and ensure that it is always visible in panels which support the command area.

The fixed command area should be at the bottom of the panel, just above the function key area (if present). Do not provide a user option to change the location of the command line.

Function key area

The *CUA Basic Interface Design Guide* discusses the function key area. This is the area at the bottom of a panel that lists available actions and their physical key assignments. Some of the actions are common dialog actions because they have common meanings in all applications.

Some other actions that appear in the function key area may be unique to the individual application.

BMS supports twenty-four function keys. The attention identifier (AID) associated with each key is available to application programs in the CICS-supplied copy-book, DFHAID. If you decide that the user may select function keys by cursor position, as well as by pressing the appropriate function key, then your application must detect the cursor position in one of the two methods described on page 15. You are responsible for the layout of the function key area, but you should observe the CUA guidelines.

Recommendation

If you know that the whole terminal environment supports 24 function keys then design your application to use all 24, but in a mixed 12- and 24-key environment, design for 12 keys only.

Scrolling panel areas

The *CUA Basic Interface Design Guide* defines scrolling for a number of situations, and BMS supports all of them.

The scrollable area consists of a number of identically formatted lines (for example, help information formatted as single field lines, or tables with data in columns). If all the scrollable data is fixed, you can define the data as a series of pop-ups. However, if the data is variable and is derived from the processing of your application, you can define a single BMS map that contains both scrollable and non-scrollable panel areas. Your application must update the scrollable part of the panel, and any scrolling information in the non-scrollable part (such as the number of lines displayed and the total number to be displayed). Display the data by issuing a SEND MAP DATAONLY command to update the panel. In this way, with a single map, you can scroll any number of lines, according to the needs of your application.

Recommendation

To minimize coding when building a scrollable panel for variable data, redefine the working storage output data area as an array. If you do this, you must ensure that when you make any changes to the map you also change the array and recompile the application. For an example of this technique, see the copybook DFH0BLST, where LSTY redefines LSTI.

Chapter 6. BMS application design for the CUA entry model

Many CICS BMS applications tend to be data-entry intensive, with the only available actions being *enter data*, or *browse*. These types of application are well suited for a CUA entry model, because with limited actions available, they maximize the resources of the host system by minimizing the number of host interrupts.

Using minimum function BMS, you can use all of the CUA entry model requirements as described in the *CUA Basic Interface Design Guide*. Note, however, that although the entry model example described there shows a pop-up for the prompt function, pop-ups for dialogs are optional and a full screen could have been used.

Recommendation

Use the entry model in all cases where a major redesign of the application is not required.

Do not use pop-ups in an entry-level application.

© Copyright IBM Corp. 1989,1999

Chapter 7. BMS application design for the CUA text model

To illustrate the recommendations discussed in the preceding chapters, CICS provides a sample application to demonstrate BMS support for CUA. The BMS support for CUA in this application uses object-action programs with action bars, pull-downs, and pop-ups. The application programs demonstrate how to code COBOL/VSE programs to display, overlay, and remove CUA style windows. They demonstrate most of the CUA panel display combinations, together with the rebuild and removal combinations that occur in a real application.

The CUA text model application programs are based on the *threshold 3* level discussed in “Designing the user interface” on page 9, and develop the use of CUA principles to browse, add, update, and delete records in a customer file.

The application performs elementary error checking to show where this can occur, but the main objective is to show what can be implemented, and the amount of design effort that is involved. All menu choices are activated although not all the routes through the sample are complete (for example, the print option is stubbed). This is because the missing routes would simply duplicate CUA features and program functions demonstrated elsewhere in the application. Another reason for omitting some routes is that the application could not be supplied in a state ready for running because of dependencies such as printer ids. However, you can take this model and enhance it to meet your own requirements by referring to the *CUA Basic Interface Design Guide*. You can also use it to evaluate BMS support for CUA and the recommendations made earlier in this book.

The CUA text model application programs present a CUA appearance and use a common action bar with associated pull-downs, pop-ups, and help panels, all working in a similar way to that which you would experience in CUA interface techniques used in any other application.

This chapter describes the sample application in three main sections, giving firstly an explanation of some design decisions used in the sample, followed by two overviews of the application. The first overview is from the user’s perspective, the second from the viewpoint of an application designer. The three sections are:

1. Some application design considerations
2. The end-user’s view
3. The designer’s view.

Some application design considerations

The CUA text model application uses an object-action approach throughout, together with standard CUA terminology where appropriate. There is only one object, which is the customer data file. Our user scenario assumes that the terminal end-user has already selected the customer file from a list of other objects, and is now ready to “open” the file to do one or more of the following:

- Browse the contents
- Update one or more customer records
- Add new customer records
- Delete customer records
- Print customer information

© Copyright IBM Corp. 1989,1999

- Select help panels.

There are many ways in which this application could be written, using CUA guidelines, to achieve the same end user function. The following design considerations apply to this sample application:

- The dialog flow is not optimized for any particular user.
- The application closely follows the CUA sample text subset example in the *CUA Basic Interface Design Guide*, and adapted to the application database requirements.
- The application uses a common action bar for all panels. It uses only the *file* and *help* actions because the application does not allow different *views*, or *user options*.
- To avoid illogical user situations, some actions in the file pull-down are automatically deactivated at some points, especially in the action list processing.
- The application uses mixed case for all data.
- The ENTER key acts as an accelerator key to take the user directly from the primary panel to the *open browse* pop-up.
- The F3 function key exits from the primary panel to CICS, but returns the user to the primary panel from any other panel in the application.
- The application does not use a command line, but reserves line 23 in every map. You can use this line as a command line if you want to enhance the sample in this way.
- The F1 help function displays contextual help in a pop-up to the right-hand side of each panel. The application fills the help pop-up with relevant information from a help file.
- Elementary record locking is done to prevent two terminals updating the same record. This is achieved by writing the terminal ID to the last 4 bytes of the record while it is in use.

List processing design

There are many application decisions which you must make about how to process an action list.

The CUA text model application uses the following techniques:

- When the list panel is displayed, the user continues the dialog by typing single character action codes against the customer records of interest. The letters 'B', 'U', and 'D' are valid in this application.
- There is a '+' field that shows there is more data to be seen if the user presses F8. The application shows this field as '- +' if scrolling is possible in both directions.
- When actions are typed against several records, an action list is generated and stored in a TS queue (LISTtrmid). When the user presses ENTER, this list is processed sequentially.
- If a displayed list is not what the user wants, the user can "step back" to the initial *open* dialog using the F12 (cancel) function, and modify the original selection criteria.

- When the user opens the file and is viewing a list, the application deactivates the ‘open’ action in the file pull-down.
- Any use of F3 (exit) cancels the action list processing.
- Any user branch to a base panel other than those being processed by an action list, result in the termination of list processing.
- While processing an action list, navigation forward from one full panel to the next will be done by F5=Next (thus leaving the Enter key for panel verification purposes), navigation backward will be done by F12, namely, Cancel back through panels processed by an action list.
- The application dynamically changes function keys in list panels and only displays them when they are available to the user. The application does not close up the blank space left by function keys that are not available on a particular display. (An alternative is to superimpose an asterisk (*) over the first character of an unusable function key and leave it displayed; the programming technique is similar, and it is a matter of choice.)

The list processing actions described above are shown in Figure 3 on page 32.

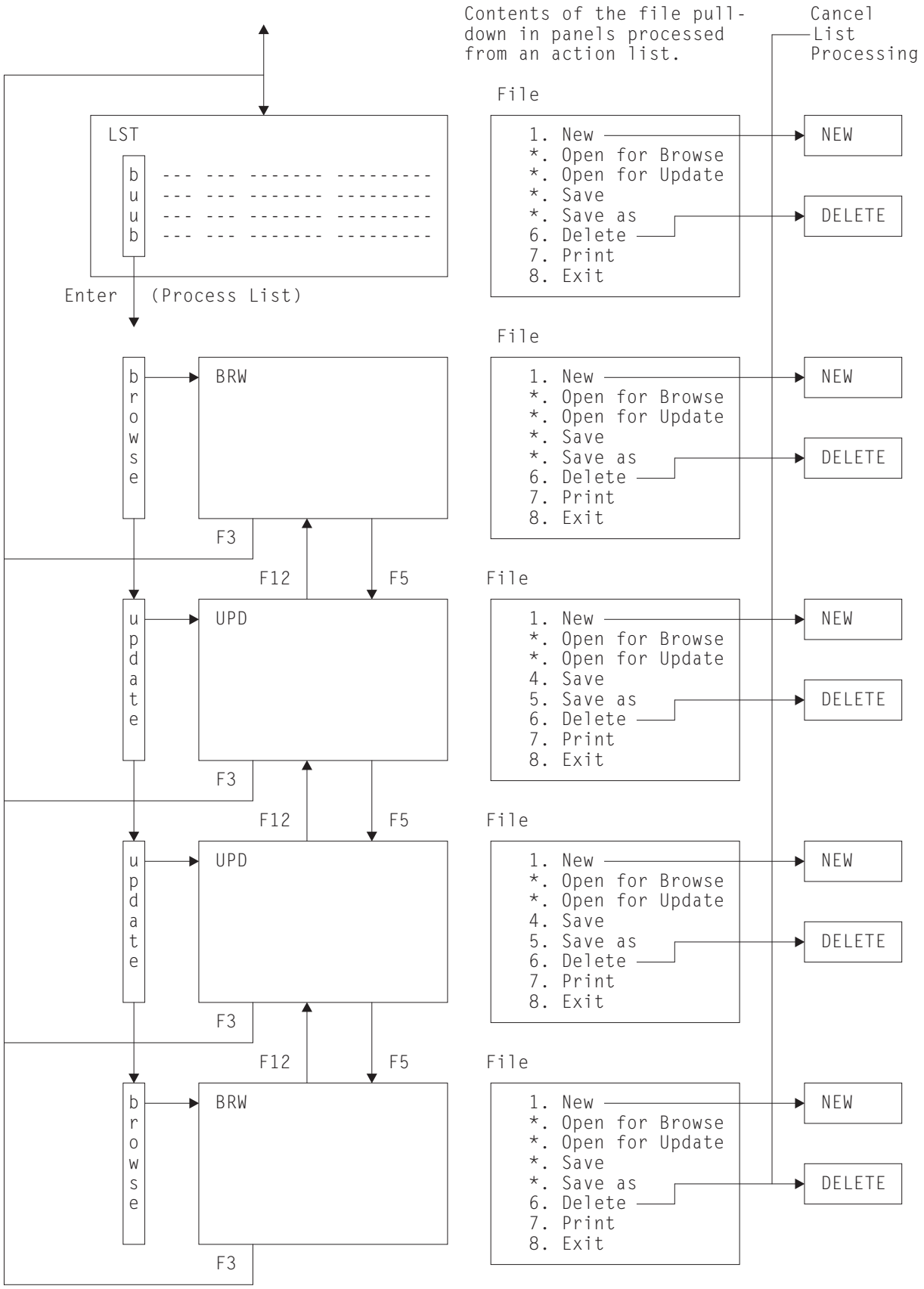


Figure 3. The CUA sample application text model list processing

The end-user's view

This section describes the CUA text model application from the end-user viewpoint, giving an overview of the user interface and interactions.

This is a basic application, designed to convey the general flavor of the CUA text subset. It is not feasible to show all the combinations of user interactions, and therefore only some of the flows through the application have been selected to illustrate the style of the user interface, and the main user interactions. Where applicable, the overview includes brief notes on significant CUA points, to highlight the techniques mentioned earlier in this book.

If you have access to the CUA text model application you can try the programs for yourself, as well as exploring paths that are not described here. Running the application should show the maps in the CUA colors. The BMS maps are correctly coded for CUA, but they might be displayed differently if you are running under an emulator.

Throughout this description, the panels are referred to by their title and the panel identifier, which appears in the top left-hand corner.

Starting

Assuming that you have a CICS region running, the application is correctly installed, and you have a clear screen in front of you, type in the transaction code AC20. Note that although all the programs have their own transaction codes, AC20 is the only valid entry, because all the other programs run pseudo-conversationally from this.

Panel T1 is the primary panel that the application displays.

```
  File  Help
--------------------------------------------------------------------------------
T1                            Customer Data File









M0001 (C) Copyright IBM Corporation 1991.  All rights reserved.

F1=Help  F3=Exit  F10=Actions
```

Notes

In a production environment, users would typically select the customer data file option (panel T1) from an initial menu that a controlling application displays when users sign on. The CUA text model application design assumes that this is the case, and T1 is the primary menu for this application.

The normal user interaction is to switch to the file action in the action bar by either pressing F10 or moving the cursor with the tab or cursor keys.

If the user presses an incorrect key, for example F7, then an error message appears on the message line.

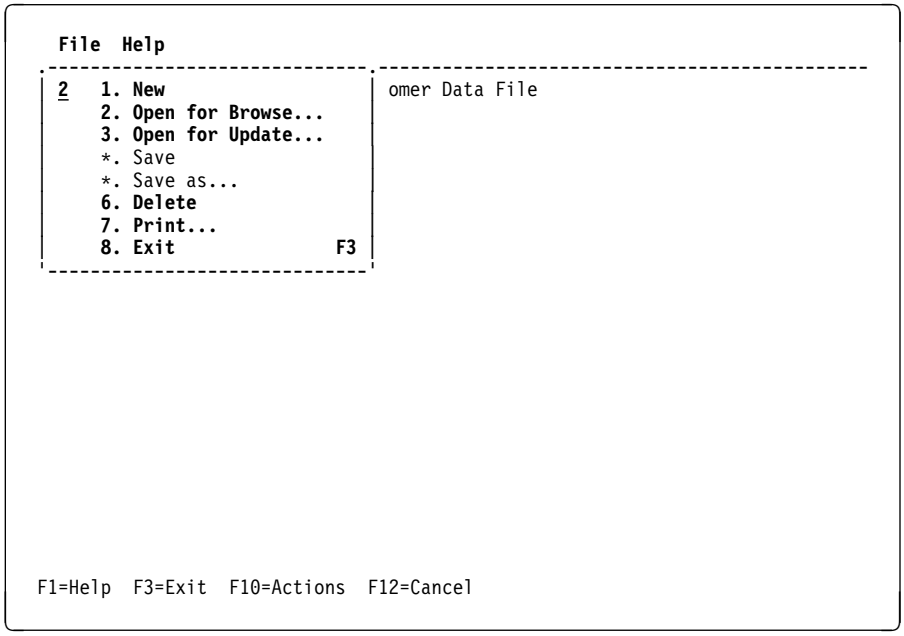
For the purpose of this overview, assume the user presses F10 to switch the cursor to the file action in the action bar, followed by the ENTER key to display the file pull-down.

CUA considerations

- This is the base panel.
- The separator line is drawn with hyphens.
- The copyright message is displayed only once, the first time the application is invoked.
- There are no special application or BMS considerations in displaying this panel.

Choosing what to do in the file pull-down

The application displays the *file pull-down* illustrated here in response to the “File” action selected in panel T1.



Notes on the file pull-down

Selections 4 and 5 are not available because there is nothing to save at this point.

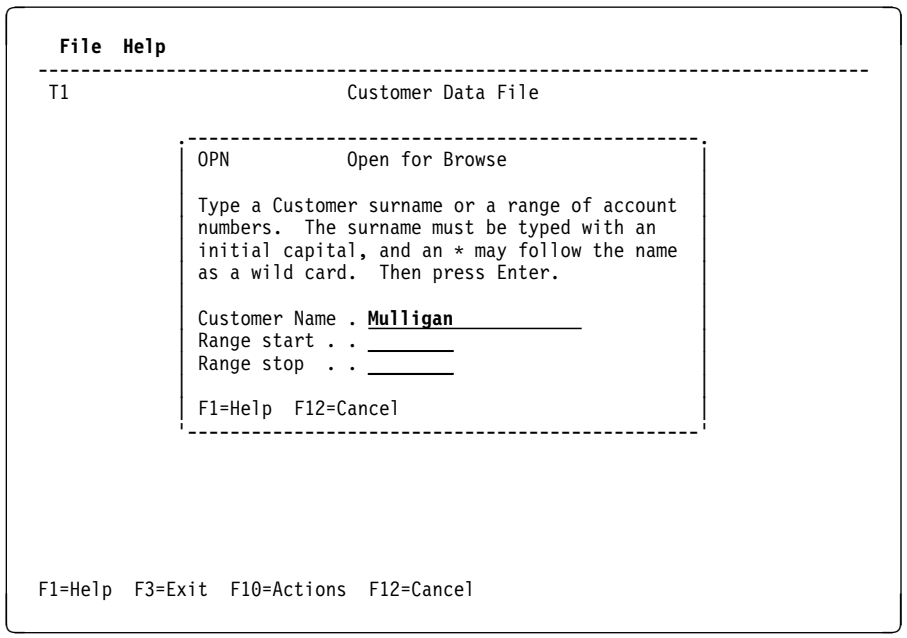
For the purpose of this overview, assume the user wants to open the customer data file to see a specific customer record. The user types a 2 in the selection entry field (the ellipses indicate that a pop-up continuation dialog follows), and presses the ENTER key.

CUA considerations

- The T1 panel is still the base panel.
- The pull-down box outline is drawn by hyphens, vertical bars, periods, and apostrophes.
- The file pull-down panel (FPD) is overlaid on the base panel without the base panel being erased.
- The copyright message is removed.
- The application highlights and protects the file action.
- The application de-emphasizes the ‘save’ and ‘save as’ actions and replaces the associated selection numbers with asterisks.
- The application must rebuild the base panel and the pull-down if the user presses the CLEAR key.

The open pop-up with specific search criteria

The application displays the *Open for Browse* pop-up (OPN) in response to the 2 entered in the selection entry field in the file pull-down on the previous panel.



Notes on the open pop-up with specific search criteria

The user can also reach the open pop-up panel (OPN) by using ENTER as an accelerator key from the primary panel T1.

If any input data is incorrect the application displays a message using the single message line, and emphasizes the appropriate input fields.

For the purpose of this overview, assume the user types a correct entry of "Mulligan" in the customer name field and presses ENTER.

Alternatively, the user could:

- Enter a specific account number on the range start line, in which case the application would display the browse panel (BRW).
- Enter a start and stop range of account numbers, in which case the application would display a list panel (LST) similar to the partial search case discussed later.

CUA considerations

- The T1 panel is still the base panel, and is rebuilt to remove the file pull-down overlay.
- The use of a pop-up here is an application choice. From a CUA viewpoint it is equally acceptable to display a full screen panel. The pop-up is a small map overlaid (without erase) on the T1 panel.
- Only the function keys inside the pop-up are valid.
- The application must rebuild the base panel and the pop-up if the user presses the CLEAR key.

Browse customer panel

The application displays the *Browse Customer* panel in response to the specific customer name entered in the open pop-up on the previous panel. This is shown in the following sample screen.

```
  File  Help
--------------------------------------------------------------------------------
BRW                            Browse Customer


Customer details

 Account Number  : 00000010
 Surname . . . . : Mulligan
 First Name  . . : Gerald
 Address . . . . : 23, St. James Street
 Town  . . . . . : Portsmouth
 County  . . . . : Hampshire
 Postcode  . . . : PO56 3PO
 Credit Limit  . : 3500
 Account Status  : A
 Comments  . . . : Good Customer
                   Sometimes pays on time
                   Not reliable




F1=Help  F3=Exit           F10=Actions  F12=Cancel
```

Notes on the browse customer panel

Because the user entered a specific name for which a record existed in the file, the application displays the customer details.

From this panel F3 returns the user to the primary panel T1 while F12 returns to the open pop-up, OPN.

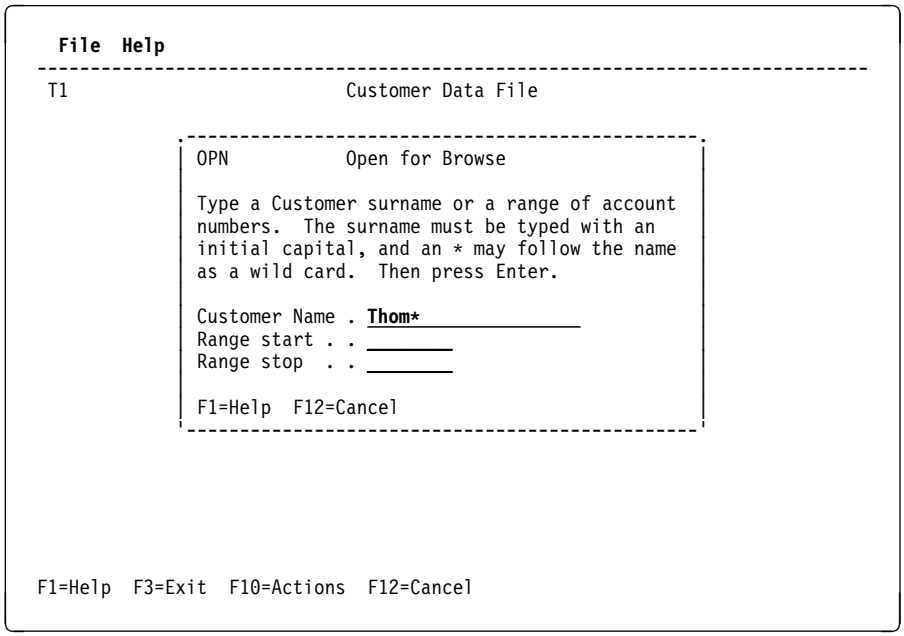
This overview continues by assuming the user presses F12 again to return to the previous panel (OPN), and changes the search criteria to 'Thom*', using the asterisk to represent any trailing letters.

CUA considerations

- This is now the base panel.
- The function key line is dynamically changed to allow for F5=Next. Because this is not operational when the browse panel (BRW) is viewed directly from the open pop-up (OPN), space is reserved for it since it would be shown when this panel is viewed from an action list.

The open pop-up with general search criteria

This overview now returns to the *Open for Browse* panel, in response to F12 on the previous panel.



Notes on the open pop-up with general search criteria

This overview continues by assuming the user enters a partial name, using an asterisk (*) to represent any letters, then presses ENTER.

CUA considerations

- The panel is handled the same as for the specific entry case, except that a list will result should multiple records qualify. If only one record is found, the result is identical to the specific case.

The list panel

The application displays the *Customer List* (LST0) panel in response to the partial name entered in the customer name field in the open pop-up.

```
 File  Help
--------------------------------------------------------------------------------
LST0                        Customer List

Type one or more action codes then press Enter.

B=Browse  U=Update  D=Delete

                                          Items     1 to     4 of     4

Action       Account No  Surname                First Name

  b           00000030   Thomas                 Alan
  u           00000005   Thompson               Chris
  u           00000006   Thompson               Cindy
  b           00000007   Thomson                Simon




F1=Help  F3=Exit          F8=Fwd  F10=Actions  F12=Cancel
```

Notes on the list panel

In response to the partial name with an asterisk representing any final letter, more than one record met the qualifying criteria; therefore the application displays a list (LST0) panel with any qualifying records. Note that the panel shown here represents what will be displayed with the supplied file data. If anyone has made any changes, additions, or deletions to the data file then the actual number of records you see displayed may vary. You can also reach a similar list panel by entering both a range start and a range stop on the OPN panel.

The actions of browse, update and delete are available from the list by typing a single character action code against the required customer record.

When you select a number of records, the application queues them ready for display, and you can work on them sequentially. In this application, F3 terminates the current display, any queued records waiting for display, and returns you to the primary T1 panel. Using F3 for this purpose is an application choice.

This overview continues by assuming the user types actions of b, u, u, and b in the action fields of the first four records displayed. The dialog from this point follows that shown in Figure 3 on page 32. Full screen customer details panels are shown according to the order of the action list generated.

CUA considerations

- This is now the base panel.
- The function key line is a variable because, in this case, F7=Bkwd is not available when the list is first displayed, but is after you scroll the list forward.
- List panels are the most complex in application programming terms, because of the amount of data you must keep available for re-display purposes when users scroll forward and backward. However, this has nothing to do with CUA – list panel handling is just as complex in non-CUA applications.
- When the action list is completed, the action characters are replaced by asterisks (*) to show the actions were processed, and any messages are placed on the line adjacent to the record they reference.

Browse customer panel from an action list

The application displays the *Browse Customer* (BRW) panel in response to the 'b' entered against the first customer in the list of the previous panel.

```
  File  Help
--------------------------------------------------------------------------------
BRW                            Browse Customer


Customer details

 Account Number  : 00000030
 Surname . . . . : Thomas
 First Name  . . : Alan
 Address . . . . : 16, Roman Road
 Town  . . . . . : Streatham Vale
 County  . . . . : London
 Postcode  . . . : SW16
 Credit Limit  . : 5000
 Account Status  : A
 Comments  . . . : Good Customer
                   Pays on time
                   Reliable




F1=Help  F3=Exit  F5=Next  F10=Actions  F12=Cancel
```

Notes on the browse customer panel from an action list

From this panel:

- F3 returns you directly to the primary panel (T1)
- F12 returns you to the list in the previous panel (LST0) cancelling any further action list processing.

ENTER is not a valid key to page forward through the action list.

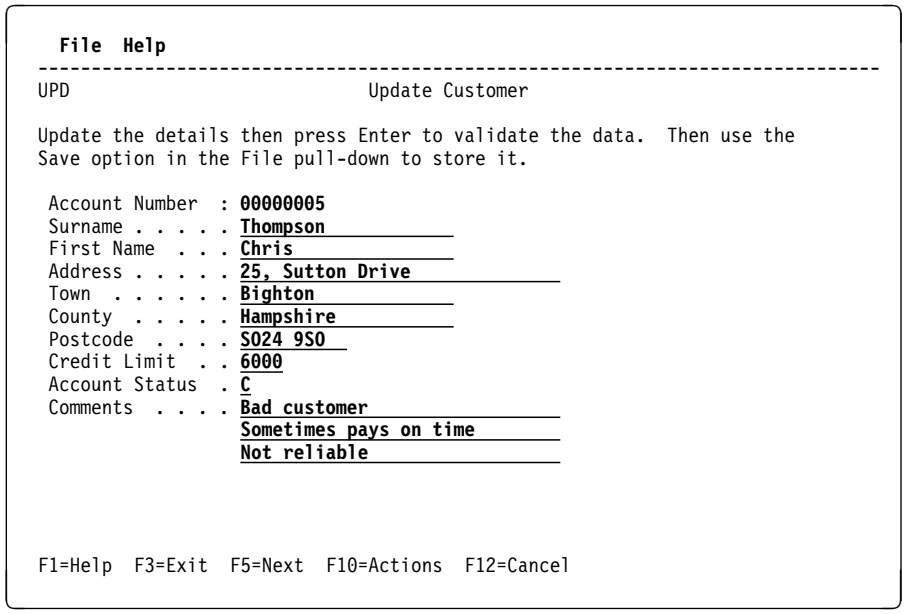
Assume the user continues with the action list and presses F5=Next, to continue.

CUA considerations

- This is now the base panel.
- F5=Next is now available because this instance of the browse panel is for a customer selected from an action list, unlike the previous browse example which was displayed for a specific customer selected by name in the open pop-up (OPN). (See pages 35 and 37).
- The use of F5 for **Next** in the dialog flow is an application choice. Although F5 is assigned to **Refresh** by CUA, refresh is not used in this application and we are therefore free to use the key for our own purposes.

Update customer panel from an action list

After the browse panel from the action list, the application displays the *Update Customer* (UPD) panel in response to the 'u' entered against the second customer in the list panel (shown earlier).



Notes on the update customer panel from an action list

From this panel:

- F3 returns you directly to the primary panel (T1)
- F12 returns you to the previous panel in the action list.

The ENTER key is now available for verification of any data that the user changes in the panel.

This overview continues by assuming the user changes the second comment to "Getting better all the time," then presses the ENTER key.

CUA considerations

- This is now the base panel.
- If the user wants to view any previous panels, F12=Cancel steps back through the previous action list panels.

Update customer panel after verification

The application responds to the ENTER key following the change by overlaying a message on the panel as shown in the following sample screen.

```
  File  Help
--------------------------------------------------------------------------------
UPD                             Update Customer

Update the details then press Enter to validate the data.  Then use the
Save option in the File pull-down to store it.

 Account Number  : 00000005
 Surname . . . . . Thompson
 First Name  . . . Chris
 Address . . . . . 25, Sutton Drive
 Town  . . . . . . Bighton
 County  . . . . . Hampshire
 Postcode  . . . . SO24 9SO
 Credit Limit  . . 6000
 Account Status  . C
 Comments  . . . . Bad Customer
                   Getting better all the time
                   Not reliable


M0014 Data validated - use Save in the File pull-down, or F12 to Cancel.

F1=Help  F3=Exit  F5=Next  F10=Actions  F12=Cancel
```

Notes on the customer update panel after verification

When the data is validated, the user presses F10, which moves the cursor to the file action, then presses the ENTER key.

This overview continues by assuming the user presses F10, then the ENTER key to obtain the *file* pull-down.

CUA considerations

- There are no CUA or BMS considerations in displaying this panel.

Saving a customer record

The application responds to the file action request by overlaying the file pull-down on the update customer panel.

```
  File  Help
.------------------------------.-------------------------------------------
| 4   1. New                   | date Customer
|     *. Open for Browse...    |
|     *. Open for Update...    | er to validate the data.  Then use
|     4. Save                  | to store it.
|     5. Save as...            |
|     6. Delete                |
|     7. Print...              | _____
|     8. Exit...          F3   | _____
'------------------------------' ive
 Town  . . . . . . Bighton
 County  . . . . . Hampshire
 Postcode  . . . . SO24 9SO
 Credit Limit  . . 6000
 Account Status  . C
 Comments  . . . . Good Account
                   Getting better all the time
                   Not reliable



F1=Help  F3=Exit  F4=Previous  F5=Next  F10=Actions  F12=Cancel
```

Notes on saving a customer record

To save the data, the user types 4 in the selection entry field and presses the ENTER key. Although not shown in this overview, the application responds by displaying a message indicating that the data is saved. The dialog with the action list continues when the user presses F5 to display the next record. The user then steps through the remaining actions.

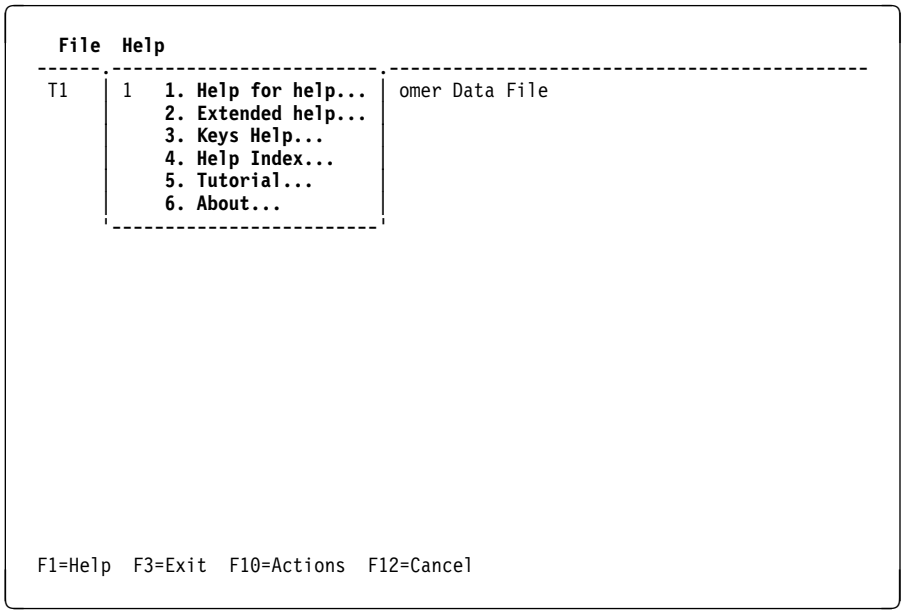
CUA considerations

- The program controls the dialog according to the diagram shown in Figure 3 on page 32. Note that the application must be written to control any situations that might occur if the user selects items from pull-downs while they are still processing an action list.
- The “Open for Browse” and “Open for Update” actions are deactivated because the file is already open.
- The “Save” and “Save As” actions are re-activated and the numbers restored.
- The user may try to save from the pull-down before the data in the underlying update panel has been verified. The application must take steps to prevent this.

This concludes the general overview of the sample application. The following text explains various other panels available and the “stubbed” exits that would require further expansion in a fully-functioning application.

Choosing what to do in the help pull-down

The application displays the *help pull-down* (HPD), if the user selects it from the Action Bar.



Notes on the help pull-down

If the user selects the help pull-down (HPD), various options are available. Only the “6. About” option works in this application, the remaining options 1 to 5 all leading to a common help panel (HLP). A real application would need to provide the correct links to appropriate help panels as required.

For the purpose of this discussion, assume the user types a 1 in the selection entry field.

CUA considerations

- The help pull-down (HPD) is handled exactly the same as the file pull-down (FPD) except that it leads to a common program stub, which is used for options 1 to 5.

The help stub

The application displays the help *stub* panel in response to the user selecting 1 on the help pull-down.

```
HLP                                Help

An application would implement help according to its requirements.  The
option you selected in the Help pull-down was followed by ellipses and
therefore the user would expect a pop-up to follow.  This panel is treated
as a full screen pop-up for the purposes of the sample program.  The
following specific pop-ups could be implemented:

1.  Help for Help - This information tells users how to get help and how
                    to use the help facilities
2.  Extended Help - This information tells users about the tasks that
                    can be performed in the application panel
3.  Keys Help     - A list of the application keys and their assignments
4.  Help Index    - A list of the help information available for the
                    application
5.  Tutorial      - Access to a tutorial if the application provides one
6.  About         - Access to the copyright and ownership information

F12=Cancel
```

Notes on the help stub

This is the help stub panel (HLP). The only possible user response in this sample application is F12 (Cancel).

CUA considerations

- This is an example of a full screen pop-up. Any underlying panels are not rebuilt if Clear is pressed because they will never be seen.

Contextual help

```
 File  Help
_   1. New                    Data File
    2. Open for Browse...
    3. Open for Update...
    *. Save
    *. Save as...
    6. Delete
    7. Print...
    8. Exit              F3

                                          Help: File Selection

                                          1=Create a New customer.
                                          2=Browse one or more
                                            customer details.
                                          3=Update one or more
                                            customer details.
                                          4=Save New or Updated
                                            customer details.
                                          5=Save an existing customer
                                            detail as a new account.
                                          6=Delete a customer detail.
                                          7=Print as required.

                                          F12=Cancel

F1=Help  F3=Exit  F10=Actions  F12=Cancel
```

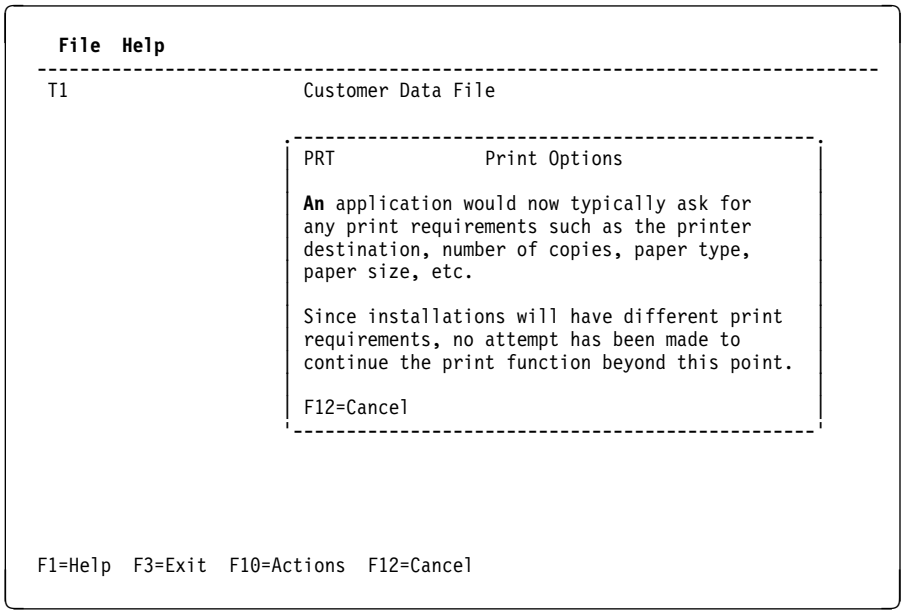
Notes on contextual help

This is just one example of contextual help. The user presses F1 with the cursor positioned on the entry field of the file pull-down. The help pop-up (HPOP) appears on the right-hand side of the panel and it contains help material, taken from the help file, that is relevant to the cursor position.

CUA considerations

- The help pop-up (HPOP) is handled the same as any other pop-up. When the pop-up is removed (by either F12 or accidental Clear), the application must rebuild any maps below it (in this example, T1 and FPD, the file pull-down).

The print stub



Notes on the print stub

This is just an example of a print pop-up (PRT). To get here, the user types a 7 in the entry field of the file pull-down (FPD) and the pop-up is overlaid over any base panel.

Like the help function in this sample application, the print panel is handled by a print stub program. The only possible user response is F12 (Cancel).

CUA considerations

- The print pop-up (PRT) is handled the same as any other pop-up. When the pop-up is removed (by either F12 or accidental clear), the application must rebuild any maps below it, (in this example, T1).

This completes the user's view of the CUA text model application. There are many routes and interactions, not all of which can be covered here. The programmer is recommended to explore the application and the CUA interactions by installing it on a CICS system. Although it is not possible for a 3270-type terminal to behave exactly like a PS/2 workstation, a CUA text subset application such as this should present few surprises to an end-user familiar with a CUA interface.

The designer's view

The design of the CUA text model application is based on a structured, modular method to make efficient use of code, and to avoid duplication wherever possible. Common programs are established wherever they can be used as sub-routines (for example, to display and process the pull-downs from the action bar). Some are common external routines (for example, the overlay, file handling, and error procedures). You can apply much of this coding technique to any new CUA application.

If the application needs to rebuild a panel, it transfers control to the program that displayed the panel initially.

The sample application 'help' function is coded in a common program, and can operate on a field-detectable basis. Although it only operates in some areas of the application, the design enables you to adapt it for use on any panel.

All file I/O is handled by common programs, which perform the user database accesses. The main program calls the file I/O programs as required. The purpose of having all the file I/O in special programs is to separate the dialog and display code from the database access code. There are two file I/O modules: (1) for the main customer data file, and (2) for the help file. Each of these files can be defined to CICS as either local or remote files, but usually you would define the help file as local.

The transactions and programs correspond to the panels as follows:

Table 2. Summary of the CUA text sample programs

Transid	Program	Map id	Mapset	Description
AC20	DFH0VT1	T1	DFH0T1	Primary panel
AC21	DFH0VOL	FPD	DFH0FPD	Pull-down
		HPD	DFH0HPD	Pull-down
AC22	DFH0VOPN	OPN	DFH0OPN	Open
AC23	DFH0VLST	LST	DFH0LST	List
AC24	DFH0VNEW	NEW	DFH0NEW	New
AC25	DFH0VBRW	BRW	DFH0BRW	Browse
AC26	DFH0VUPD	UPD	DFH0UPD	Update
AC27	DFH0VDEL	DEL	DFH0DEL	Delete
AC28	DFH0VPRT	PRT	DFH0PRT	Print
AC29	Not Used	N/A	N/A	N/A
AC2A	DFH0VSAS	SAS	DFH0SAS	Save As
AC2B	Not used	N/A	N/A	N/A
AC2C	DFH0VHLP	HLP	DFH0HLP	Help stub
AC2D	DFH0VAB	AB	DFH0AB	Abend
AC2E	DFH0VHP	HPOP	DFH0HP	Help pop-up
AC2F	DFH0VABT	ABT	DFH0ABT	About
N/A	DFH0VRIO	N/A	N/A	Remote file I/O (customer data)
N/A	DFH0VLIO	N/A	N/A	Local file I/O (help file)
N/A	DFH0VTBL	N/A	N/A	Table router
DELQ	DFH0VDQ	N/A	N/A	Delete TS queues

Panel processing is dependent on indicators set in either the communication area or temporary storage queues. Because of the number of paths a user can take through the application, it is not possible to generate a simple step-by-step flow diagram to show all the program routes. Typical user routes only are shown, and the diagram in Figure 4 on page 50 shows the general relationship of the individual programs that make up this sample application.

Resource usage

Based on the threshold levels described under “Designing the user interface” on page 9, it is possible to give some general guidance on some aspects of application design that you should consider in relation to existing installation standards and practices. It is important to remember that many of the factors affecting efficiency are caused by application design choices, and not because of CUA guidelines. For example, within the CUA text model application, should the end user be allowed to use a function key as a means of a fast path between browse and update? The application logic may become more complex and add to the overhead, but give greater freedom for the end user. The choice depends upon

the end-user, or installation, requirements. Using entry level, for new or existing applications, probably involves little or no overhead.

Using text level, and giving the user flexibility to use the action bar as a fast-path, and using pop-ups, you need to use temporary storage to “remember” panels and paths through the user’s dialog.

From such information the application must be able to re-display the previous sequence of panels, including any user data. You, as the application designer, might want to build in some restriction on data and panel recall, since the user could choose a sequence of actions that would keep on using temporary storage. For example, in action lists, if the user were to repeatedly use the file pull-down to branch around a list and open further lists (without returning to the main menu), an increasing amount of temporary storage would be used to record the flow. Temporary Storage is deleted as soon as it is no longer required. Previously created TS queue entries that become redundant are reused.

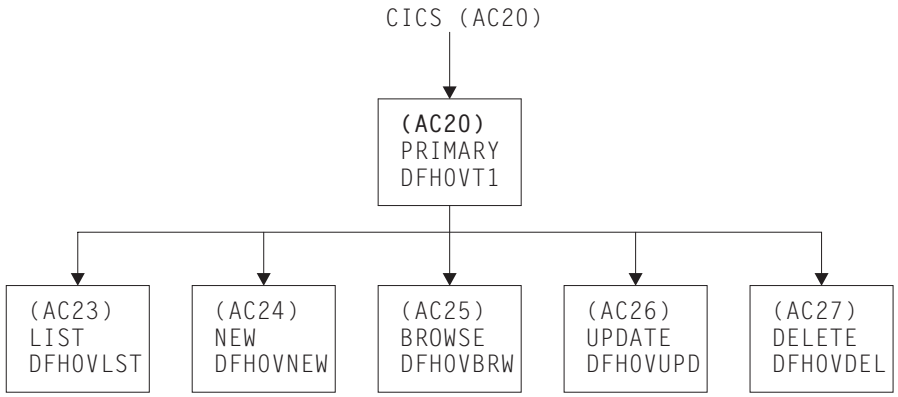
As soon as an action bar is present, an associated “pull-down” window is required to proceed further. This can be regarded as a miniature full screen panel which will probably increase the level of dialog management required. In the environment of a non-programmable terminal you must consider the additional data flow between the host system and the terminals; in particular, if the terminals are connected remotely then the additional response times may be unacceptable. Balanced against this must be the added usability gained by the use of action bars, function keys, colors, and so on, that relate consistently to specific functions, and the increased flexibility given to the overall system design.

Pop-ups are similar to pull-downs. Where a pop-up or pull-down needs to be removed from the user display in order to complete the required screen image, the underlying base panel must be refreshed before any subsequently required pop-ups/pull-downs can be displayed.

For example, the CUA text model application requests customer names or numbers by use of an “open” pop-up over the primary panel, which appears after the user selects it from the “file” pull-down. To clear the file pull-down, the primary panel must be re-sent, followed by the open pop-up.

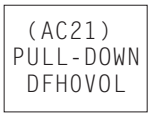
Program relationships

The application has a primary base panel (T1) which can route (via pop-up and utility programs) to five further base panels as shown in Figure 4.



Any of the above base panels can have pop-ups superimposed to allow the user to move between them.

The program controlling the Action Bar pull-down displays is:



The programs controlling the pop-up displays are:

The routing between the programs and the actual file I/O is controlled by the following utility programs:

Figure 4. CUA text model application program relationships

Notes:

1. DFH0VTBL, DFH0VOL, DFH0VAB, DFH0VRIO, and DFH0VLIO are common programs that could be initiated from any panel directly.
2. DFH0VRIO is a common program that performs all the reading and writing to the customer data file, which can be situated remotely.
3. DFH0VLIO is a common program that performs all the reading and writing to the help file, which can be situated locally.

Program structure

The general structure of the CUA text model application programs is as follows:

Initiation

Initiation is performed from an initial transaction, AC20, which displays the primary (T1) panel. All other programs are invoked either directly as a result of a CICS RETURN, an XCTL (transfer of control), a LINK, or a dynamic COBOL call from another program. Indicators in the communication area are tested,

giving the program knowledge of how it was called, and also the type of map being received. Different processing is applied to base, pop-up, or pull-down maps, allowing for alternative function keys to be activated, and attributes to be reset in the panel displayed.

Process

If entry is via an XCTL – process XCTL valid function keys.

If entry is from CICS – receive map and process valid function keys.

Process the ENTER key according to user input, cursor position, or indicators in the communications area and/or temporary storage (TS) queues.

Process invalid function keys.

Note: The CLEAR key requires specific handling since there is no way to prevent it clearing a 3270 screen.

Routing

The multiple routes available through the application and the control of the action bar pull-down selections are set out in a three-dimensional table array, which is accessed via a table program. This enables the program routing to be determined according to the action, base panel, and selection. If multiple objects were required, a fourth dimension could be added as a higher level of this table. As there is no direct routing control in any module, existing modules are not affected if new ones are added.

Termination

Display appropriate panel, return next transaction code, or CALL/XCTL as required.

Commarea

The communication area is used during processing to maintain status indicators, counters, and so on. It contains fields to indicate:

- * The pull-down displayed indicator
- * The base panel displayed indicator
- * The pop-up panel displayed indicator
- * The action selected indicator
- * The browse/update action mode
- * The panel type
- * The TS queue counter for referencing panel images
- * The TS queue counter for referencing stored records
- * The current entry in the record index
- * The program entry state indicator
- The CICS response field
- The call number for error processing
- The help field key
- The I/O call type
- The I/O call return code
- * The counter to process the record index
- The pull-down selection number
- The next program name to pass control to
- The action index storage field
- The base index storage field
- The selection index storage field
- The search level required indicator
- The action not available indicator (returned from search)

- The selection not available indicator (returned from search)
- The selections processed counter field
- The option selected indicator field
- The customer Account Number
- The range start number
- The range stop number
- The customer name
- The last item read from the list TS queue
- * The number of list panel items selected for processing
- * The number of list panel items processed
- * The list TS queue item number processed by the selected actions
- * The list panel line number processed by the selected actions
- The total number of items that met the search criteria
- The contextual help cursor position hold field
- The list panel displayed/processing indicator
- The panel confirm/validated indicator
- The terminal ID hold area, for message build processing only.

Note: In the CUA text model application, the communication area size is 200 bytes but 61 bytes of this is filler for expansion purposes. Only the items marked with ***** are unique to the CUA requirements.

TS queues

The CUA text model application uses temporary storage queues during processing to pass data between transactions/programs. (TRMID is the terminal address from where the transaction was started.)

The following temporary storage queues are used:

- A panel queue (PANLtrmid). This queue stores the BMS output panel images.
- A *record-hold* TS queue (RECDtrmid). The first item of this TS queue is reserved to keep track of where the user actually is in the application. It is this item which indexes the PANL and RECD TS queues. There may be up to 50 records in this entry, which stores the following data:

  Panel name (N)
  The panel name can be a base panel name or a selection number indicating a pull-down selection.

  Panel type (T)
  If a pop-up is displayed, the panel type is:

  'b' for base panel
  'f' for full screen pop-up, or
  'p' for pop-up.

  Action (A)
  The action is the action which was selected from the action bar prior to this base or pop-up panel.

  Panel item number (I)
  The panel item number is a pointer to the entry in the PANL TS queue that contains the panel image.

Record item number (R)

The record item number is a pointer to the entry in the RECD TS queue which contains the record relating to this panel image.

The item numbers following the first entry contain the actual customer file records for processing by the application.

The structure and relationship of the PANL and RECD TS queues is shown in Figure 5.

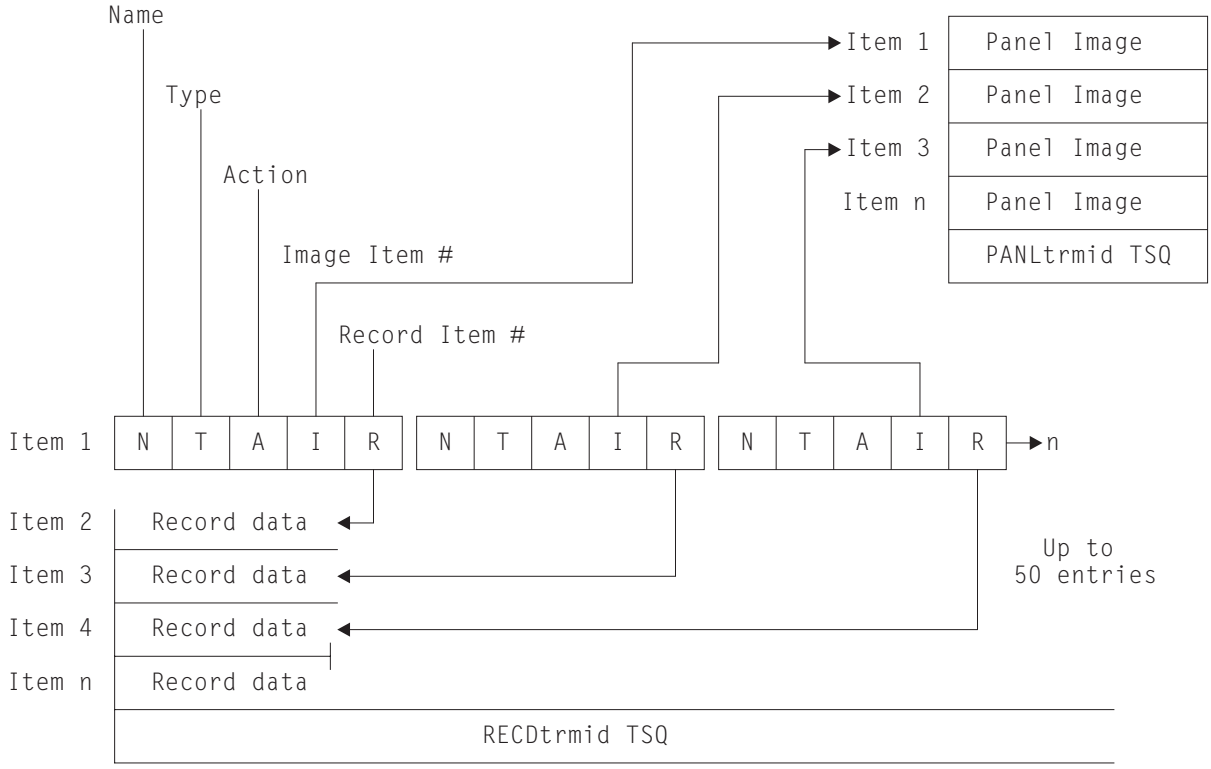


Figure 5. Structure and relationship of PANL and RECD TS queues

- A customer data file I/O queue (LISTtrmid). This queue is only written if more than one record meets the search criteria. This list queue only contains those customer details that are necessary to format the list panel, that is, account number, surname, and first name. This optimizes temporary storage usage.
- A help file I/O queue (HELPtrmid). This queue is written when the help file is read to supply contextual help to fill the help pop-up.

Cleaning up TS queues

When the application runs normally, any TS queues are purged correctly. If further development work is done on this application and an abend occurs during testing, there can be a small nuisance factor when using CEBR to access and purge TS queues generated relevant to the Terminal Net-ID. For this reason there is a temporary storage clean-up transaction supplied, called DELQ. It purges all the temporary storage queues used by the application and frees any records that are held for update.

The program associated with this transaction is called DFH0VDQ.

Chapter 8. Installing and running the CUA text model application

This chapter describes what you need to do to install and run the CUA text model application programs. There are four sections in this chapter:

1. Generating the BMS maps
2. Translating, compiling, and link-editing the application programs
3. Creating the VSAM files
4. Installing and running the application on your CICS region, including modifying CICS-supplied resource definitions to allow mixed-case input.

Generating the BMS maps

The CUA text model application programs, maps, and copy books are all in PRD1.BASE. Before you can translate and compile the programs, you must first generate the maps. You need to generate all the maps shown in Table 3. Note that the member names are of the form DFH0Mxxx, but the corresponding mapsets are named DFH0xxx. This permits you to write each symbolic description map (copy book) back to the same library as the source member.

Table 3. Summary of CUA sample application maps

Map / Map set	Member	Description
AB / DFH0AB	DFH0MAB	General purpose map for abend handling
ABT / DFH0ABT	DFH0MABT	Information about the sample application
BRW / DFH0BRW	DFH0MBRW	Browse customer details
DEL / DFH0DEL	DFH0MDEL	Delete a customer record
FPD / DFH0FPD	DFH0MFPD	File pull-down
HLP / DFH0HLP	DFH0MHLP	The help (stub) panel
HP / DFH0HP	DFH0MHP	Contextual help panel
HPD / DFH0HPD	DFH0MHPD	Help pull-down
LST / DFH0LST	DFH0MLST	List processing panel
NEW / DFH0NEW	DFH0MNEW	New customer record
OPN / DFH0OPN	DFH0MOPN	File-open panel
PRT / DFH0PRT	DFH0MPRT	Print panel
SAS / DFH0SAS	DFH0MSAS	Save changed customer record
T1 / DFH0T1	DFH0MT1	Primary panel to sample application
UPD / DFH0UPD	DFH0MUPD	Update and validate customer details

Translating, compiling, and link-editing the application programs

When you have successfully generated the maps into suitable libraries, you can translate and compile the application programs. The programs are listed in Table 2 on page 48.

© Copyright IBM Corp. 1989,1999

Creating the VSAM files

The CUA text model application requires two VSAM key-sequenced data sets; (1) the customer details file, and (2) the help file. You should define these to VSAM and load them with the CICS-supplied initial data. To define and initialize these data sets, run the jobs DFH0DCUS and DFH0DHLP. These jobs can be found in PRD1.BASE and must be tailored to suit your CICS environment.

```
// JOB DEFCUASF CREATE VSAM CUA SAMPLE CLUSTER
/* *================================================================*
/* * This job prepares the CUA Sample VSAM files.                   *
/* *                                                               *
/* * It issues IDCAMS DELETE commands to delete the ALTERNATEINDEX  *
/* * and CLUSTER definitions for the VSAM files to make sure they   *
/* * do not exist before issuing an IDCAMS DEFINE CLUSTER to define *
/* * the VSAM data file.                                           *
/* *                                                               *
/* * It then executes the DFH$CUAD phase to load the defined data. *
/* * The data is read from 80 byte SYSIPT records and the data from *
/* * three records are combined to produce a single 227 byte output *
/* * record which is written to the VSAM file. To identify the three *
/* * input records, the first byte is a either 1, 2 or 3 which      *
/* * defines which part of the output record the data is for.       *
/* * If the input record's sequence numbers are validated to ensure *
/* * that each output record is made up of 3 input records. If the  *
/* * sequence is broken, the input records are ignored up to the    *
/* * next input record with its first byte set to 1.               *
/* *                                                               *
/* * The next step issues an IDCAMS DEFINE ALTERNATEINDEX to define *
/* * the Alternative Index VSAM entry. This is then linked to the   *
/* * data file (base cluster) with the IDCAMS DEFINE PATH command.  *
/* *                                                               *
/* * Finally the Alternate Index is built by issuing the IDCAMS     *
/* * BLDINDEX command and the results listed by the IDCAMS LISTCAT  *
/* * command.                                                      *
/* *                                                               *
/* * Note: You must change all occurences of the following fields:  *
/* *                                                               *
/* *       user_catalog                                            *
/* *       volume                                                  *
/* *================================================================*
/*
```

Figure 6 (Part 1 of 5). Job to define and initialize the customer details file

```
/* *******************************************************************
/* * Delete the VSAM entries for the Alternate Index and Cluster if  *
/* * they already exist and then define the Cluster.                 *
/* *******************************************************************
// EXEC IDCAMS,SIZE=AUTO
 DELETE                                                 -
          (DFHCTPTH)                                    -
          PATH                                          -
          CATALOG(user_catalog)
 DELETE                                                 -
          (CICS410.SAMPLE.DFH0FAI)                      -
          ALTERNATEINDEX                                -
          CATALOG(user_catalog)
 DELETE                                                 -
          (CICS410.SAMPLE.DFH0FUS)                      -
          CLUSTER                                       -
          CATALOG(user_catalog)
 DEFINE CLUSTER                                         -
          (NAME(CICS410.SAMPLE.DFH0FUS)                 -
           INDEXED                                      -
           TRACKS (2 1)                                 -
           BUFFERSPACE(4096)                            -
           RECORDSIZE(227 227)                          -
           KEYS(8 0)                                    -
           SHAREOPTIONS(2)                              -
           VOLUMES(volume))                             -
         DATA                                           -
           (NAME(CICS410.SAMPLE.DFH0FUS.DATA))          -
         INDEX                                          -
           (NAME(CICS410.SAMPLE.DFH0FUS.INDEX))         -
         CATALOG(user_catalog)
/*
/* *******************************************************************
/* * Now load the VSAM Cluster with data
/* *******************************************************************
// DLBL USERCAT,'user_catalog',,VSAM
// DLBL CUADATA,'CICS410.SAMPLE.DFH0FUS',,VSAM,CAT=USERCAT
// LIBDEF *,SEARCH=(PRD1.BASE,PRD2.SCEEBASE)
// EXEC DFH$CUAD
100000001Gardner             Derek               1, Main Street               P
2ortsmouth          Hampshire           PO12 3CT  3500AGood Customer
3     Sometimes eats               Oranges for lunch
100000002Graham              Eric                2, West Way                  W
2ickham             Hampshire           SO31 7HI  3000AGood Customer
3     Pays on time                 Reliable
100000003George              Harry               2718 Anderson Mill Drive     O
2wlesbury           Hampshire           SO67 6RM  4000BPoor Customer
3     Sometimes pays on time       Not reliable
100000004Gregory             Mike                378, Algarve Terrace         F
2our Marks          Hampshire           SO99 4IR  5000AGood Customer
3     Pays on time                 Reliable
```

Figure 6 (Part 2 of 5). Job to define and initialize the customer details file

```
100000005Thompson            Chris               25, Sutton Drive             B
2ighton             Hampshire           SO24 9SO  6000CBad Customer
3     Sometimes pays on time        Not reliable
100000006Thompson            Cindy               67, Shawford Close           B
2ramdean            Hampshire           SO15 9TB  7000AGood Customer
3     Pays on time                  Reliable
100000007Thomson             Simon               35, Appledown Crescent       H
2ambledon           Hampshire           SO24 9HI  8000AGood Customer
3     Pays on time                  Reliable
100000008Tomson              Roger               19, Park Road                W
2inchester          Hampshire           SO24 9ON  1000AGood Customer
3     Pays on time                  Reliable
100000009Robinson            Tim                 Houseboat 3, Hamble Marina   H
2amble              Hampshire           SO87 2MS  1000CBad Customer
3     Never pays on time            Very unreliable
100000010Mulligan            Gerald              23, St.James Street          P
2ortsmouth          Hampshire           PO56 3PO  3500AGood Customer
3     Sometimes pays on time        Not reliable
100000011Graham              Ernest              2, West Way                  W
2inchester          Hampshire           SO65 7SN  3000AGood Customer
3     Pays on time                  Reliable
100000012George              William             217, Central Avenue          A
2lton               Hampshire           SO84 6OW  4000BPoor Customer
3     Sometimes pays on time        Not reliable
100000013Beckett             Roger               36, Orchard Grove            R
2opley              Hampshire           SO93 4NR  5000AGood Customer
3     Pays on time                  Reliable
100000014Spencer             Fred                12A, Hobart Street           W
2est Meon           Hampshire           PO35 2IO  3500AGood Customer
3     Sometimes eats                Scones for tea
100000015Graham              Chris               Surridge Farm                L
2ymington           Hampshire           BM46 2ST  3000AGood Customer
3     Pays on time                  Reliable
100000016Williams            Fiona               21, Chalgrove Road           C
2hilbolton          Hampshire           SO05 1GE  4000BPoor Customer
3     Sometimes pays on time        Not reliable
100000017Zaventem            Oscar               14, East Street              A
2lresford           Hampshire           SO94 0RT  5000AGood Customer
3     Pays on time                  Reliable
100000018Williams            Lynda               54, London Road              B
2ishops Sutton      Hampshire           SO11 0EH  5000AGood Customer
3     Pays on time                  Reliable
100000019Woolliams           Stephanie           Lower Wield Farm             G
2undleton           Hampshire           SO32 4AI  5000AGood Customer
3     Pays on time                  Reliable
100000020Westaway            Tony                87, Lanham Close             N
2orthington         Hampshire           SO99 6TS  5000AGood Customer
3     Pays on time                  Reliable
```

Figure 6 (Part 3 of 5). Job to define and initialize the customer details file

```
100000021Becker              John                46, Orchard Grove            G
2uildford           Surrey              SO93 4II  5000AGood Customer
3     Pays on time                Reliable
100000022Sparks              Philip              167, Hobart Street           D
2orking             Surrey              PO35 2NN  3500AGood Customer
3     Sometimes eats              Bread for supper
100000023Grant               Colin               97, The Rise                 G
2atwick             Surrey              BM46 2HH  3000AGood Customer
3     Pays on time                Reliable
100000024Wilton              Frances             51, Chiltern Avenue          C
2ranleigh           Surrey              SO05 1UU  4000BPoor Customer
3     Sometimes pays on time      Not reliable
100000025Ziegler             Jan                 77, West Street              M
2itcham             Surrey              SO94 0RR  5000AGood Customer
3     Pays on time                Reliable
100000026Winston             Hilda               54, Mitcham Road             C
2roydon             Surrey              SO11 0SS  5000AGood Customer
3     Pays on time                Reliable
100000027Williamson          Stanley             34, Nursery Road             H
2ackbridge          Surrey              SO32 4LL  5000AGood Customer
3     Pays on time                Reliable
100000028West                Jim                 22, Vermont Close            C
2arshalton          Surrey              SO99 6EE  5000AGood Customer
3     Pays on time                Reliable
100000029Fox                 Tom                 61, Windermere Road          S
2treatham Vale      London              SO99 6YY  5000AGood Customer
3     Pays on time                Reliable
100000030Thomas              Alan                16, Roman Road               S
2treatham Vale      London              SW16      5000AGood Customer
3     Pays on time                Reliable
100000099Test99              99
2                                                    ZTest record only
3
100000999Test999             999
2                                                    ZTest record only
3
100009999Test9999            9999
2                                                    ZTest record only
3
100099999Test99999           99999
2                                                    ZTest record only
3
100999999Test999999          999999
2                                                    ZTest record only
3
109999999Test9999999         9999999
2                                                    ZTest record only
3
199999999ABCDEFGHIJKLMNOPQRSTAbcdefghijklmnopqrst1234567890ABCDEFGHIJKLMNOPQRSTA
2BCDEFGHIJKLMNOPQRSTABCDEFGHIJKLMNOPQRSTABCDEFGHIJ9999ZAaaaaaaaaaaaaaaaaaaaaaaa
3aaaaaBbbbbbbbbbbbbbbbbbbbbbbbbbbbbbCccccccccccccccccccccccccccc
/*
```

Figure 6 (Part 4 of 5). Job to define and initialize the customer details file

```
/* *================================================================*
/* * Define the Alternate Index, and the Path.                      *
/* *================================================================*
// EXEC IDCAMS,SIZE=AUTO
 DEFINE ALTERNATEINDEX                                  -
          (NAME(CICS410.SAMPLE.DFH0FAI)                 -
           RELATE(CICS410.SAMPLE.DFH0FUS)               -
           NONUNIQUEKEY                                 -
           UPGRADE                                      -
           BUFFERSPACE(4096)                            -
           RECORDSIZE(200 200)                          -
           RECORDS(5 5)                                 -
           FREESPACE(20 10)                             -
           KEYS(20 8)                                   -
           SHAREOPTIONS(2)                              -
           VOLUMES(volume))                             -
         DATA                                           -
           (NAME(CICS410.SAMPLE.DFH0FAI.DATA))          -
         INDEX                                          -
           (NAME(CICS410.SAMPLE.DFH0FAI.INDEX))         -
         CATALOG(user_catalog)
 DEFINE PATH                                            -
          (NAME(DFHCTPTH)                               -
          PATHENTRY(CICS410.SAMPLE.DFH0FAI))            -
          CATALOG(user_catalog)
/*
/* *================================================================*
/* * Finally, build the Alternate Index and then issue LISTCAT     *
/* * commands to check everything is ok.                            *
/* *================================================================*
// DLBL IJSYSUC,'user_catalog',,VSAM
// DLBL DFH0FUS,'CICS410.SAMPLE.DFH0FUS',,VSAM,CAT=USERCAT
// DLBL DFH0FAI,'CICS410.SAMPLE.DFH0FAI',,VSAM,CAT=USERCAT
// EXEC IDCAMS,SIZE=AUTO
 BLDINDEX                                               -
           INDATASET(CICS410.SAMPLE.DFH0FUS)            -
           OUTDATASET(CICS410.SAMPLE.DFH0FAI)           -
           CATALOG(user_catalog)
 LISTCAT ENTRIES(CICS410.SAMPLE.DFH0FUS)ALL             -
         CATALOG(user_catalog)
 LISTCAT ENTRIES(CICS410.SAMPLE.DFH0FAI)ALL             -
         CATALOG(user_catalog)
/*
/&
```

Figure 6 (Part 5 of 5). Job to define and initialize the customer details file

```
// JOB DEFHELP CREATE VSAM CUA SAMPLE HELP FILE
/* ================================================================== */
/* * This job consists of two steps. The first defines the VSAM files * */
/* * needed for the CUA Sample Applications Help file. The second     * */
/* * stage loads the help information into the VSAM file.             * */
/* *                                                                  * */
/* * Change the "user_catalog" to the name of your VSAM User Catalog  * */
/* * and "volume" with the label of a disk with VSAM space.           * */
/* ================================================================== */
// EXEC IDCAMS,SIZE=AUTO
 DELETE                                                 -
          (CICS410.SAMPLE.DFH0FLP)                      -
          CLUSTER                                       -
          CATALOG(user_catalog)
 DEFINE CLUSTER                                         -
          (NAME(CICS410.SAMPLE.DFH0FLP)                 -
           CONTROLINTERVALSIZE(512)                     -
           RECORDSIZE(38 38)                            -
           TRACKS(2 1)                                  -
           KEYS(10 0)                                   -
           SHAREOPTIONS(2)                              -
           VOLUMES(volume))                             -
        DATA                                            -
          (NAME(CICS410.SAMPLE.DFH0FLP.DATA))           -
        INDEX                                           -
          (NAME(CICS410.SAMPLE.DFH0FLP.INDEX))          -
        CATALOG(user_catalog)
 LISTCAT ENTRIES(CICS410.SAMPLE.DFH0FLP)   ALL          -
        CATALOG(user_catalog)
/*
/* ================================================================== */
/* * Now populate the Help file with the data contained below for     * */
/* * by the sample application. You will need to change the two DLBLs * */
/* * with the catalog and file information for the file.              * */
/* *==================================================================*/
// DLBL USERCAT,'user_catalog',,VSAM
// DLBL DFH0FLP,'CICS410.SAMPLE.DFH0FLP',,VSAM,CAT=USERCAT
// LIBDEF *,SEARCH=(PRD1.BASE,PRD2.SCEEBASE)
// EXEC DFH$CUAH
        01Help: Undefined Field
        02No Help text is currently
        03available for this field.
        04
        05To get Help, the application
        06would have to define the
        07field to the program and the
        08Help Table, and add the text
        09to the Help file.  The
        10program would have to be
        11re-compiled and the Help
        12file reinstalled.
```

Figure 7 (Part 1 of 4). Job to define and initialize the help file

```
ACCNO   01Help: Account Number
ACCNO   02The account number is a
ACCNO   03maximum of 8 digits.
ACCNO   04
ACCNO   05The customer file in this
ACCNO   06sample initially contained
ACCNO   07account numbers from 1 to
ACCNO   0830 and some test data
ACCNO   09numbered 99, 999, 9999
ACCNO   10and 99999999. If you
ACCNO   11cannot access a particular
ACCNO   12account it may have been
ACCNO   13deleted or updated.
ACCST   01Help: Account Status
ACCST   02Account Status is a single
ACCST   03character code that shows
ACCST   04the customer ordering
ACCST   05authorisation required.
ACCST   06
ACCST   07A=No check needed
ACCST   08
ACCST   09B=Supervisor
ACCST   10
ACCST   11C=Manager
ACTION  01Help: Action
ACTION  02A single character action
ACTION  03code may be entered against
ACTION  04one or more customers. This
ACTION  05will generate an action list
ACTION  06which will be processed
ACTION  07sequentially. The list may
ACTION  08not be updated fully until
ACTION  09it is re-displayed.
ADDR    01Help: Address
ADDR    02The address is a maximum of
ADDR    0330 characters and may be
ADDR    04mixed case.
COMM    01Help: Comments
COMM    02There are three lines of
COMM    03comments which may be used
COMM    04for additional information.
COMM    05These lines are optional.
COUNT   01Help: County
COUNT   02The county is a maximum of
COUNT   0320 characters and may be
COUNT   04mixed case.
```

Figure 7 (Part 2 of 4). Job to define and initialize the help file

```
CRLIM   01Help: Credit Limit
CRLIM   02The credit limit is a
CRLIM   03maximum of 4 digits. It
CRLIM   04shows the maximimum amount
CRLIM   05of credit available.
FFLD    01Help: File Action
FFLD    02This action allows users to
FFLD    03work with the customer data
FFLD    04in the file.  You may create
FFLD    05New data, Browse, Update,
FFLD    06Delete, and Print the data
FFLD    07either individually or from
FFLD    08a list.
FFLD    09
FFLD    10Press F12 to return to the
FFLD    11Action and then Enter to
FFLD    12display the pull-down menu.
FNAME   01Help: First Name
FNAME   02The first name is a maximum
FNAME   03of 20 characters and may be
FNAME   04mixed case.
FPDSEL  01Help: File Selection
FPDSEL  021=Create a New customer
FPDSEL  032=Browse one or more
FPDSEL  04  customer details.
FPDSEL  053=Update one or more
FPDSEL  06  customer details.
FPDSEL  074=Save New or Updated
FPDSEL  08  customer details
FPDSEL  095=Save an existing customer
FPDSEL  10  detail as a new account.
FPDSEL  116=Delete a customer detail.
FPDSEL  127=Print as required.
HFLD    01Help: Help Action
HFLD    02This action allows users to
HFLD    03access various kinds of help
HFLD    04information.
HFLD    05
HFLD    06Press F12 to return to the
HFLD    07Action and then Enter to
HFLD    08display the pull-down menu.
HPDSEL  01Help: Help Selection
HPDSEL  021=Information about Help.
HPDSEL  032=Information about the
HPDSEL  04  contents of the panel.
HPDSEL  053=A list of function keys
HPDSEL  06  and their assignments.
HPDSEL  074=A list of help available.
HPDSEL  085=An application tutorial.
HPDSEL  096=Copyright and application
HPDSEL  10  version information.
```

Figure 7 (Part 3 of 4). Job to define and initialize the help file

```
PCODE   01Help: Postcode
PCODE   02The postcode is a maximum of
PCODE   0320 characters and may be
PCODE   04mixed case.
RSTART  01Help: Range Start
RSTART  02If an account number is
RSTART  03typed in range start without
RSTART  04a higher number in range
RSTART  05stop then a single customer
RSTART  06detail will be displayed.
RSTOP   01Help: Range Stop
RSTOP   02If an account number is
RSTOP   03typed in range stop and it
RSTOP   04is higher than range start
RSTOP   05then a list of all the
RSTOP   06customers in the required
RSTOP   07range will be displayed.
SAS     01Help: Save As
SAS     02The next available account
SAS     03number is automatically
SAS     04allocated by the system.
SAS     05This is an application
SAS     06choice to prevent saving
SAS     07over an existing account.
SNAME   01Help: Surname
SNAME   02The surname is a maximum of
SNAME   0320 characters.  It MUST be
SNAME   04typed with an initial
SNAME   05capital to be consistent
SNAME   06with the existing data in
SNAME   07the file.  Mixed case
SNAME   08searching is not a function
SNAME   09of this sample program.
TOWN    01Help: Town
TOWN    02The town is a maximum of
TOWN    0320 characters and may be
TOWN    04mixed case.
T1TITLE 01Help: Customer Data File
T1TITLE 02There is no help available
T1TITLE 03at this point in the sample
T1TITLE 04application.  If the program
T1TITLE 05was fully developed, then
T1TITLE 06further information about
T1TITLE 07the application would be
T1TITLE 08available via extended help
T1TITLE 09in the Help pull-down.
T1TITLE 10Press F12 to return to the
T1TITLE 11main panel, then F10 to get
T1TITLE 12to the Actions or Enter to
T1TITLE 13display the Open Pop-up.
/*
/&
```

Figure 7 (Part 4 of 4). Job to define and initialize the help file

Installing and running the application on your CICS region

The resource definitions you need to run the CUA text model application are supplied in the CSD, either when you initialize a new CSD or upgrade an existing one. The CUA text-level resource definitions are in a CICS-supplied group named DFH$CTXT. You can either use the CEDA INSTALL command to install the group on your running CICS region, or you can add the DFH$CTXT group to your start-up group list and perform a cold start. However, in addition to installing the sample application group, you must also modify some of the CICS-supplied groups before you are ready to run the application. This is because the application is designed to use mixed-case input. To enable this to work correctly, you must modify the uppercase translation attributes on a number of CICS-supplied resource definitions.

Logmode

In order to get the correct colors displayed, you must make sure that you log on with a full 7-color VTAM® logmode. For programming information, including a list of queriable logmodes, see the CICS Transaction Server for VSE/ESA Customization Guide.

Modifying resource definitions to support mixed-case input

If your terminal definitions (autoinstall models or otherwise) reference the CICS-supplied typeterm definitions, CICS translates all input from your terminal into uppercase. This is because the typeterm definitions specify UCTRAN(YES), and this takes precedence over the uppercase translation option on profile resource definitions, where the default is UCTRAN(NO) on the CICS-supplied definitions. This automatic translation of terminal input to uppercase is unsuitable for the CUA text model application and would result in “record not found” messages occurring unexpectedly.

The CUA text model application, and particularly the customer data, is designed to be used in mixed-case mode. Therefore, before attempting to run the application, you should switch off uppercase translation for your terminal, and use translation selectively at the transaction level.

Switching off uppercase translation at the terminal level

To switch off uppercase translation for your terminal, identify which model typeterm definition it uses, and change UCTRAN(YES) to UCTRAN(NO). If your terminal is autoinstalled, and you are in doubt about which typeterm definition your terminal is installed under, look for message number DFHZC6935 in the CADL transient data queue. This is usually sent to the MSGUSR data set, and directed to SYSOUT. The following is an example of this message, which is preceded by DFHZC5966:

```
DFHZC5966 I 09/26/94 13:07:04 CICSIDA INSTALL started for TERMINAL   ( S010 )
                                      ( Module name: DFHBSTZ ).
DFHZC6935 09/26/94 13:07:04 CICSIDA Autoinstall for terminal : S010,
                  NETNAME IGKS010 , using model-name DFH3270, successful
```

In this example, the typeterm definition is the CICS-supplied DFH3270. If the typeterm definition referenced by your terminal is installation-defined, you may be able to make the change without any difficulty. However, if your test system shares its CSD with other CICS test regions, you should ensure that you make the change in a way that does not affect other users. If necessary, you should create terminal and typeterm definitions specifically for your own terminal so that you can run the CUA text model application.

Unlocking and altering IBM-protected definitions

If the typeterm definition referenced by your terminal is one of the CICS-supplied definitions, you cannot alter attributes directly. To alter a CICS-supplied definition, which is IBM-protected, you must first copy the resource to a group of your own. You can use the following CEDA commands to copy and alter a typeterm definition, and replace the CICS-supplied group with your altered group in your CICS start-up group list:

```
CEDA COPY   GROUP(DFHTYPE)  TO(userTYPE)

CEDA ALTER  TYPETERM(type_name) UCTRAN(TRANID) GR(userTYPE)

CEDA REMOVE GROUP(DFHTYPE)  LIST(usrLIST)

CEDA ADD    GROUP(userTYPE) LIST(usrLIST)
```

Changing the UCTRAN attribute to TRANID causes CICS to translate to uppercase only the transaction id (the first four characters entered at the terminal). This avoids you having to define lowercase aliases for all the CICS-supplied transactions, and ensures that when you type transaction names such as CEMT, CEDA, and CESF, CICS recognizes them. The remainder of terminal input after the first four characters is translated according to the UCTRAN attribute on the transaction profile definition (DFHCICST for CICS-supplied transactions). However, DFHCICST specifies UCTRAN(NO), which causes difficulties for CICS-supplied transactions that require many operands in uppercase. For example, CEMT INQUIRE FILE(filex) fails with the NOT FOUND condition if the installed file is called FILEX. To ensure that uppercase translation is performed on all the data you type on CICS-supplied transactions (so that they work as they did before you switched off translation at the terminal level) change DFHCICST to specify UCTRAN(YES).

Changing a CICS profile definition for uppercase translation

To change the translation attribute on the CICS-supplied profile DFHCICST, first copy the profile to a new group, and then change the UCTRAN parameter. For example, you can use the following CEDA commands to copy and alter the profile definition, and replace the CICS group with your modified group in the start-up group list.

```
CEDA COPY   GROUP(DFHSTAND) TO(usrSTAND)

CEDA ALTER  PROFILE(DFHCICST) UCTRAN(YES) GR(usrSTAND)

CEDA REMOVE GROUP(DFHSTAND) LIST(usrLIST)

CEDA ADD    GROUP(usrSTAND) LIST(usrLIST)
```

When all the resource definitions are modified and installed as you need them, you are ready to run the application from a suitable CICS terminal.

Security

Ensure that the sign-on userid you are using is suitably authorized to access the CUA text model application and the associated resources.

Chapter 9. CUA text model program descriptions

This chapter gives an overview of each of the CUA text model application programs in the form of a short summary of what each one does, highlighting any technical points and specific CUA features. The programs are written in COBOL.

Program DFH0VT1 – primary panel

This program is invoked by transaction AC20. It displays the primary panel, T1, and performs all associated processing.

It is initiated by the user entering the AC20 transaction code. All subsequent interaction is either via the transaction code or via a transfer of control from another module.

At the start of this program the Communications Area length in the EIB is examined to see if it is zero. If it is, the user has just entered 'AC20' to invoke the application. The module initializes the COMMAREA, sets up and sends the initial panel with the cursor in position, writes the initial entries to the tracking and panel TS queues and returns control to CICS with AC20 as the next tranid. This section of code should be executed only once for each invocation of the application.

The valid function keys within the T1 panel are:

F1 Request contextual help.

F3 Exit to CICS.

F10 Position the cursor.

F12 Exit to CICS.

Action on invocation by transfer of control

Processing is dependent upon which function key has been pressed.

When F3

If T1 is the only panel on the screen, or if a pull-down is displayed over it, the application is terminated and exits to CICS via the Good Morning Message module, DFHGMM. Otherwise the panel is sent with the cursor positioned on the main title field and control is returned to CICS with AC20 as the next tranid.

Note: If this application is run on CICS OS/2, the route to DFHGMM will result in an error since DFHGMM does not exist in that environment.

When F10

The panel TS queue entry is read, the panel is sent with the cursor positioned on the title field and control is returned to CICS with AC20 as the next tranid.

When F12

The first task is to rebuild the base panel image. This is done by reading the stored image queue (PANLtrmid TS queue) for the current entry. The next step is to position the cursor correctly. If a pull-down has yet to be displayed over this base map to complete the rebuilt image then no cursor positioning is performed. If a pull-down was displayed over the base map, the cursor is positioned on the action bar depending on which pull-down was displayed, otherwise the cursor is positioned on the title field of the main panel. After this

© Copyright IBM Corp. 1989,1999

the panel image is sent. A test is performed to see if this base panel image completes the rebuilding or if another module must be called. If processing is complete, control is returned to CICS with AC20 as the next tranid, otherwise the tracking queue is read and used to route to the next program via DFH0VTBL.

When CLEAR

Processing is the same as for F12 except that there is no need to test to see if a pull-down needs to be displayed. The reason for this is that processing CLEAR from the pull-down module executes the 'reset panel' code.

'Reset panel' processing

This is invoked from the DFH0VOL pull-down module when the user presses CLEAR, moves along the action bar to select another pull-down, or selects a pull-down option that requires the resetting of the base panel and routing to another module.

The 'reset panel' code reads the PANLtrmid TS queue to find the current panel image, which is then sent to the screen before control is passed either back to the DFH0VOL module or to the next required panel as necessary.

Action on invocation by transaction code AC20

After a return to CICS with AC20 as the next tranid, the program is reentered when the user presses a function key. The T1 map is received since contextual help is available on it and the cursor position needs to be established. There are no user updatable fields on the T1 map. The DFH0VTBL routine is called to establish the current position on the program routing table. The valid function keys are processed as follows.

When F1

The field cursor attributes are tested to determine on which field contextual help is required. The appropriate commarea parameter is set up and passed via the routing program to DFH0VHP. Before control is transferred, the current cursor position is saved in the commarea to allow for later repositioning of the cursor on the field from which contextual help was requested.

When F3

Processing exits to CICS via DFHGMM, the Good Morning Message program.

When F10

The only processing required is to determine the new position of the cursor. If it is on the action bar, it is moved to the title field, and if it is on the title field it is moved to the first action bar field ie FILE. The cursor is resent via a CONTROL SEND command, and control is returned to CICS with AC20 as the next tranid.

When F12

The module exits to CICS via DFHGMM, the Good Morning MESSAGE program.

When CLEAR

Processing is exactly the same as when entering via a transfer of control.

When ENTER

Processing is entirely dependent upon the cursor position. If the cursor is positioned on the action bar, parameters are set up and control is passed to DFH0VTBL to initiate the DFH0VOL module, which displays the appropriate

pull-down. If the cursor is displayed anywhere else on the panel then the 'FASTPATH' parameters are set up and the DFH0VTBL module transfers control to the DFH0VOPN module. It does this by driving it with the OPEN FOR BROWSE selection which would normally initiate the DFH0VOPN program.

Program DFH0VLST – list panel handler

This program is invoked by transaction AC23 and performs the LIST panel processing.

It is initiated via a transfer of control from the DFH0VOPN module and displays a list of customer details, showing each customer's first name, surname and account number. Against each name in this list is an action entry field in which the user may type one of B, U or D to Browse, Update, or Delete the details for that customer. After the initial display of the LIST panel, control is returned to CICS and the program is subsequently reentered when invoked via the AC23 transaction.

The valid function keys within this panel are:

F1 Request contextual help.

F3 Return to the T1 base panel.

F5 Process the next selected action.

F7 Page backwards through the LIST - not available at the beginning of the list.

F8 Page forwards through the LIST - not available at the end of the list.

F10 Toggle the cursor between the action bar and the main panel position (the first action entry field).

F12 Return to the previous panel.

Action on invocation by transfer of control

Function keys are processed as follows.

When F3

The transaction-related resources are cleared from the LISTtrmid TS queue and processing is performed to route back to the previous panel in the tracking queue (item one of the RECDtrmid TS queue). If the previous panel was a pop-up, the DFH0VTBL routine is called at the action and base levels to establish the current position. Otherwise control is just transferred to the previous module.

When F5

The next entry in the LISTtrmid TS queue is read and control is transferred to the required program to either browse (DFH0VBRW), update (DFH0VUPD) or delete (DFH0VDEL) the next customer's details. Program routing is carried out via the program DFH0VTBL.

When F10

The PANLtrmid TS queue entry is read to find the current screen image and the cursor is set on the main panel field (the first action entry field). The panel is sent and control is returned to CICS with AC23 as the next tranid.

When F12

The first task is to rebuild the base panel image from the current entry in the stored image queue (PANLtrmid). A routine is performed to build the detail panel from the LISTtrmid TS queue, and the PANLtrmid TS queue entry is rewritten.

Cursor positioning on this panel may be either symbolic or specific. If a return from an F1 is in process the stored EIBCPOSN field is used to give the specific numeric cursor position. Any other cursor positioning is entirely symbolic and dependent upon the commarea parameters. If a pull-down has yet to be displayed over this base map no cursor positioning is performed.

Once the cursor is in position, the panel image is sent and a test is performed to see if this base panel image completes the rebuilding or if routing must be performed to another module. If processing is complete, control is returned to CICS with AC23 as the next tranid, otherwise the tracking queue is read and used to route to the next program via DFH0VTBL. Specific processing is built in here to allow for control having to be passed to the DFH0VOL program to display a pull-down. This is done via the DFH0VTBL program, rather than directly, to maintain consistency.

When CLEAR

The CLEAR key functions in the same way as F12 except that there is no need to consider routing to the next module if a pull-down needs to be displayed over the top of this panel. The reason for this is that processing CLEAR from the pull-down module executes the 'reset panel' code.

'Reset panel' processing

This is invoked by DFH0VOL, the pull-down module, when the user presses CLEAR, moves along the action bar, or selects an option that results in a pop-up.

The 'reset panel' code reads the PANLtrmid TS queue to find the current panel, which is then sent to the screen before control is passed either back to the DFH0VOL module or to the next required panel as necessary.

A special case is dealt with in this process, to allow for the user selecting option 8 to exit from the FILE pull-down. This exit processing is the same as for F3 described above.

When ENTER

Firstly the tracking queue (item one of the RECDtrmid TS queue) is read and updated with the LIST panel details. Then the PANLtrmid TS queue is read to establish the current position. The updated entry is rewritten to the tracking queue and the initial values set for the panel building routine to be performed.

The panel image is built in three stages; firstly, the detail lines, secondly, information indicating which detail lines are displayed and thirdly, the function key area.

Up to 8 detail lines are displayed on the screen at a time. The building of the detail lines involves reading the next eight entries and moving the details from the TS queue layout to the panel fields. If the end of the list is reached before eight records are read, a routine is performed to turn off the action selection entry fields and the display fields for the unused lines on the screen.

Building the paging information involves using the positional indicators in the LISTtrmid TS queue to determine how many records to display and which panel ID to display. The + and - indicators are also built in this routine.

The function key area is built based upon which panel ID was set up during the paging information processing.

The cursor is positioned on the first action selection field in the panel, the tracking and panel TS queues are rewritten, the panel is sent, and control is returned to CICS with AC23 as the next tranid.

Action on invocation by transaction code AC23

When this program is invoked by CICS, the PANLtrmid is read to establish the current position and then the map is received. The valid function keys are processed as follows.

When F1

The field cursor attributes are tested to determine on which field contextual help is required. The appropriate commarea parameter is set up and passed via the table searching routine to the DFH0VHP program. Before control is transferred the current cursor position (EIBCPOSN) is saved in the commarea to allow for later repositioning.

When F3

Processing is the same as when entering via a transfer of control.

When F7

First the panel ID is checked to ensure that F7 is valid from the current screen and, if it is not, an error message is returned. Otherwise processing is performed to scroll back eight positions through the LISTtrmid TS queue. The panel building routine is performed as described for processing ENTER via a transfer of control, the panel image is rewritten to the PANLtrmid TS queue, the panel is sent and control is returned to CICS with AC23 as the next tranid.

When F8

First the panel ID is checked to ensure that F8 is valid from the current screen and, if it is not, an error message is returned. Otherwise processing is performed to scroll forward eight positions through the LISTtrmid TS queue. The panel building routine is performed as described for processing ENTER via a transfer of control, the panel image is rewritten to the PANLtrmid TS queue, the panel is sent, and control is returned to CICS with AC23 as the next tranid.

When F10

The only processing required is to determine the new position of the cursor. If it is currently positioned on the action bar it is moved to the first action selection field on the panel and if it is anywhere in the main panel it is moved to the first action bar field, FILE. The cursor is resent via a ‘control send’ and control is returned to CICS with AC23 as the next tranid.

When F12

The process described for F3 is performed to delete the LISTtrmid TS queue. Then the tracking queue is read (item one of the RECDtrmid TS queue), the entry for the current panel is cleared, and the previous base panel is found. Control is transferred to the required base panel routine to allow rebuilding of the image to begin.

When CLEAR

The panel is rebuilt from the latest entry in the PANLtrmid TS queue. The cursor is positioned as described for processing F12 via a transfer of control, the panel is sent and control is returned to CICS with AC23 as the next tranid.

When ENTER

If the cursor is positioned on the action bar to request the display of a pull-down, an indicator is set in the required commarea fields and control is transferred to the pull-down display module DFH0VOL via the table routing program.

If the cursor is not positioned on the action bar, the BROWSE, UPDATE and DELETE actions that have been entered are processed. This involves reading the LISTtrmid TS queue to find which records have action codes against them. The action codes are stored in case the user has entered '=' symbols to indicate repetitive action selections and the records are stored in a LISTtrmid TS queue entry for processing. The LISTtrmid TS queue entries and panel image details are synchronized. If any invalid action codes were entered, only the valid ones are stored, and an error message is displayed before the valid action codes can be processed. Once all the action codes have been validated, a routine to process the codes is initiated. This operates as described above for processing F5 via a transfer of control.

Program DFH0VNEW – new customer panel processing

This program is invoked by transaction AC24 and performs the processing to create a new record when the user has selected option 1 from the FILE pull-down.

It is initiated by a transfer of control from the pull-down module (DFH0VOL) and is subsequently reentered when transaction AC24 is invoked by CICS. There is no consideration given to the LISTtrmid TS queue in this module because new customer processing cannot be performed via the LIST panel.

The valid function keys within this panel are:

F1 Request contextual help.

F3 Return to the T1 base panel.

F10 Toggle the cursor between the action bar and the main cursor position (the surname).

F12 To return to the previous panel.

Action on invocation by transfer of control

Function keys are processed as follows.

When F3

The transaction-related resources are cleared and processing performed to route to the previous panel in the tracking queue (item one of the RECDtrmid TS queue). If the previous panel was a pop-up then the DFH0VTBL routine is called at the action and base levels to establish the current position, otherwise control is just transferred to the previous module.

Clearing the transaction-related resources involves deleting any records that are held on the customer file for 'create' processing but which have not been fully created. The RECDtrmid TS queue is read to check the current status of the record. If it is partially processed the DFH0VRIO module is called to delete it.

When F10

The panel TS queue entry is read to find the current screen image. The panel is sent with the cursor positioned on the main panel field (surname), and control is returned to CICS with AC24 as the next tranid.

When F12

If F12 is processed via a transfer of control then the first task is to rebuild the base panel image. This is done by reading the stored image queue (PANLtrmid) for the current entry.

Cursor positioning on this panel may be either symbolic or specific. If a return from an F1 is in process the stored EIBCPOSN field is used to give the specific numeric cursor position. Any other cursor positioning is entirely symbolic and dependent upon the commarea parameters. If a pull-down has yet to be displayed over this base map no cursor positioning is performed.

Once the cursor is in position, the panel image is sent and a test is performed to see if this base panel image completes the rebuilding or if another module must be called. If processing is complete, control is returned to CICS with AC24 as the next tranid, otherwise the tracking queue is read and used to route to the next program via DFH0VTBL. Specific processing is built in here to allow for control having to be passed to the DFH0VOL program to display a pull-down. This is done via the DFH0VTBL program, rather than directly, to maintain consistency.

When CLEAR

The CLEAR key functions in the same way as F12 except that there is no need to consider routing to the next module if a pull-down needs to be displayed over the top of this panel. The reason for this is that processing CLEAR from the pull-down module executes the ‘reset panel’ code.

‘Reset panel’ processing

This is invoked by DFH0VOL, the pull-down module, when the user presses CLEAR, moves along the action bar or selects an option that results in a pop-up.

Two special cases are dealt with in this processing, to allow for the user selecting option 8 (exit), or option 4 (save), from the FILE pull-down. The ‘exit’ processing is as described above for F3. The ‘save’ processing involves reading the panel TS queue to rebuild the panel image. If the record has not been successfully validated a message is sent to the screen; otherwise the RECDtrmid TS queue is updated to show that the new record has been created. The panel image is sent and control is returned to CICS with AC24 as the next tranid.

When ENTER

The tracking queue (item 1 of the RECDtrmid TS queue) is read and updated with the NEW panel details and the panel TS queue is read to establish the current position. The next step is to allocate a record on the Customer File. This means calling DFH0VRIO, the remote I/O module, to find the first unused customer account number in the file and to write a dummy record to hold that account number for this task. When control is returned from the I/O module the RECDtrmid TS queue entry is updated with a ‘partially complete’ indicator and the allocated account number is inserted into the NEW panel. The panel TS queue image is then written (or rewritten), the cursor is set on the surname field and NEW panel is displayed for the user to fill in the details.

Action on invocation by transaction code AC24

This program is reentered when transaction AC24 is invoked by CICS in response to the user pressing a function key whilst the NEW panel is displayed. The panel TS queue is read to establish the current position and then the map is received from the screen. The valid function keys are processed as follows.

When F1

The field cursor attributes are tested to determine on which field contextual help is required. The appropriate commarea parameter is set up and passed via the routing program to DFH0VHP. Before control is transferred, the current cursor position (EIBCPOSN) is saved in the COMMAREA to allow for later repositioning of the cursor.

When F3

Processing is the same as described for entering via a transfer of control.

When F10

The only processing required is to determine the new position of the cursor. If it is currently positioned on the action bar it is moved to the surname field, and if it is positioned anywhere in the main panel it is moved to the first action bar field, FILE. The cursor is resent via a 'control send' and control is returned to CICS with AC24 as the next tranid.

When F12

The same process as described for F3 is performed to free the held record entry and find the previous LISTtrmid TS queue record entry if necessary. Then the tracking queue is read, the latest entry is cleared, the previous base panel is found and control is transferred to the required base panel routine to allow the rebuilding of the image to begin.

When CLEAR

The panel is rebuilt from the PANLtrmid TS queue, the cursor is repositioned as described for processing F12 via a transfer of control and control is returned to CICS with AC24 as the next tranid.

When ENTER

If the cursor is positioned on the action bar to request the display of a pull-down, an indicator is set in the required commarea fields and control is transferred to the pull-down display module DFH0VOL via the routing program.

If the cursor is not on the action bar, the tracking queue and the appropriate RECDtrmid TS queue entries are read. If the RECORD VALIDATED indicator is set, it is reset to allow the user to alter any of the data; otherwise the data entered on the panel is validated. This nominal validation just involves checking that the record has been allocated successfully and that the surname is alphabetic. If the record does not already exist it is allocated and if the surname is alphabetic, the RECORD VALIDATED indicator is set. Whether the record is valid or not, the user data is stored on the RECDtrmid TS queue. The panel is then resent and control is returned to CICS with AC24 as the next tranid.

Program DFH0VBRW – browse customer details panel processing

This program is invoked by transaction AC25 and performs the processing necessary when a user wants to browse a particular record or is trying to access a list but has only retrieved a single record with the supplied criteria.

It is initiated via a transfer of control either from the DFH0VOPN pop-up module or from the DFH0VLST base module. After presenting the initial panel the program can be reentered from CICS via the AC25 transaction. The valid function keys are:

F1 Request contextual help.

F3 Return to the T1 base panel.

F5 Allow forward processing through a selection list if this panel was entered via a list panel.

F10 Toggle the cursor between the action bar and the main cursor position (surname).

F12 Return to the previous panel.

Action on invocation by transfer of control

When entering this program via a transfer of control, function keys are processed as follows:

When F3

Data relating to the current transaction is cleared from the TS queues and processing is performed to route to the previous panel. If the previous panel was a pop-up, the DFH0VTBL routine is called to search the program tables at action and base levels so as to establish the current position. Otherwise control is just transferred to the previous module.

When F5 or ENTER

An entry is written to the tracking queue (item one of the RECDtrmid TS queue) and the PANLtrmid queue is read to establish the current position. The tracking queue (item one of the RECDtrmid TS queue) is updated to show the latest position, and the routine to build the panel image is performed.

If this 'browse' action was initiated via a list, the required record must be found by obtaining the account number, from the LISTtrmid TS queue, and calling DFH0VRIO to read the customer file. This places the record in the RECDtrmid queue, which is where the DFH0VOPN module would have placed it if the BROWSE panel had been initiated directly. The data is moved from the record TS queue layout into the map fields ready for display.

The appropriate panel identifier is now set up in order to select the correct function key detail line. If all the items selected from the LIST panel have been processed a flag is set to indicate that the function key line displayed should not include F5, otherwise a flag is set to indicate that the function key line must include F5. The required function key line is selected and moved into the panel output field.

The panel is now ready to be rewritten to the PANLtrmid queue (or written if ITEMERR was raised when it was read). The cursor is positioned on the surname field, the panel is displayed and control is returned to CICS with AC25 as the tranid.

When F10

A pull-down can be removed from the BROWSE panel using F10. The PANLtrmid TS queue entry is read to obtain the latest panel image, the cursor is set on the surname field, the panel is sent and control is returned to CICS with AC25 as the next tranid.

When F12

The first task is to rebuild the base panel image by reading the stored image queue (PANLtrmid) for the current entry. Cursor positioning on the panel may be either symbolic or specific. If a return from F1 is in progress then the stored cursor position (EIBCPOSN), is used to give the specific numeric cursor position. Any other cursor positioning is entirely symbolic and dependent upon the COMMAREA parameters. If a pull-down has yet to be displayed over the base map to complete the rebuilt image then no cursor positioning is performed.

Once the panel image has been rebuilt it is sent to the screen and then a test is performed to see whether this base panel image completes the rebuilding of the screen. If processing is complete control is returned to CICS with AC25 as the next tranid; otherwise the tracking entry of the RECDtrmid TS queue is read and used to route to the next program via DFH0VTBL. Specific processing is built in here to allow for a pull-down needing to be displayed and control having to be passed to the DFH0VOL program. This routing is done via the DFH0VTBL program rather than directly so as to maintain consistency.

When CLEAR

The CLEAR key is processed in the same way as F12, except that there is no need to consider routing to another module if a pull-down needs to be displayed over the top of this panel. The reason for this is that the 'reset panel' code will be executed when processing CLEAR from the pull-down module.

'Reset panel' processing

This is invoked by the pull-down module, DFH0VOL, when the user presses CLEAR, moves along the action bar or selects an option that results in a pop-up.

The 'reset panel' code reads the PANLtrmid TS queue to find the current panel image, which is then sent to the screen before control is passed either back to the DFH0VOL module or to the next required panel as necessary.

Action on invocation by transaction code AC25

After a return to CICS with AC25 as the next tranid, the program is reentered when the user presses a function key. Although there are no user updatable fields on this screen a 'receive map' is performed to establish the cursor position. The permitted function keys are processed as follows:

When F1

The field cursor attributes are tested to determine on which field contextual help is required. The appropriate COMMAREA parameter is set up and passed via the table routing program, DFH0VTBL, to the HELP program, DFH0VHP. Before control is transferred, the current cursor position is saved in the COMMAREA to allow for later repositioning of the cursor on the field from which help was requested.

When F3

F3 processing is the same when reentering the program as when entering via a transfer of control.

When F10

The only processing required is to determine the new position of the cursor. If it is currently displayed on the action bar then it is moved to the surname field, or if it is already on the surname field then it is moved to the first action bar field. The new cursor position is sent via a 'control send' and control is returned to CICS with AC25 as the next tranid.

When F12

If this program is reentered with F12, either the user wants to scroll back through the items selected from a LIST panel or wants to return to the previous panel.

If the BROWSE panel was invoked via a LIST panel then the LISTtrmid TS queue is searched to find the previous panel. When it is found or the beginning of the queue is reached, the tracking queue (entry one of the RECDtrmid TS queue) is read, the entry for the current BROWSE panel is cleared, and the previous base panel is found. Control is then transferred to the required base panel routine to allow the rebuilding of the image to begin.

When CLEAR

The panel is rebuilt from the image stored in the PANLtrmid TS queue, the cursor is repositioned as described for processing F12 via a transfer of control, the panel is sent and control is returned to CICS with AC25 as the next tranid.

When ENTER

ENTER is valid only if the cursor is positioned on the action bar. If the cursor is not on the action bar, a message is returned, the alarm is sounded, and control is returned to CICS with AC25 as the next tranid.

However, if the cursor is on the action bar, ENTER is used to request the display of a pull-down. The cursor position is detected and an indicator is set in the COMMAREA fields. Control is then transferred to the pull-down display module DFH0VOL via the routing program DFH0VTBL.

Program DFH0VUPD – update customer record panel processing

This program is invoked by transaction AC26 and performs the processing when a user wants to update a particular record or is trying to access a list but has only retrieved a single record with the supplied criteria.

It is initiated via a transfer of control either from the DFH0VOPN pop-up module or the DFH0VLST base module. After presenting the initial panel, the program can be reentered from CICS via the AC26 transaction.

The valid function keys within this panel are:

F1 Request contextual help.

F3 Return to the T1 base panel.

F5 Allow forward processing through a selection list if this panel was entered via a LIST panel.

F10 Toggle the cursor between the action bar and the main cursor position (surname).

F12 Return to the previous panel.

Action on invocation by transfer of control

Processing is dependent upon which function key has been pressed.

When F3

The transaction-related resources are cleared, which means freeing any records on the customer file that are held for update processing but have not been completely updated. Then processing is performed to route to the previous panel in the tracking queue. If the previous panel was a pop-up, the DFH0VTBL routine is called at the action and base level to establish the current position; otherwise control is just transferred to the previous module.

If the update is being processed via a LIST panel, the LISTtrmid TS queue is read to obtain the record key, the remote I/O module (DFH0VRIO) is called to make the appropriate record available and the LISTtrmid TS queue is searched to find the next entry for processing. If the update is not processed via a LIST panel, the RECDtrmid TS queue is read to check the current state of the record, since if the update is complete no freeing is necessary. The DFH0VRIO module is called to perform the appropriate actions on the record.

When F5 or ENTER

If the record being processed has been returned to the UPDATE panel after the SAVE AS option has been processed, the panel is rebuilt from the PANLtrmid TS queue. If the record was not validated successfully before the attempt to save it then an error message is displayed; otherwise the successful completion message is built and control is returned to CICS with AC26 as the next tranid. If the SAVE AS panel was not previously processed, the PANLtrmid TS queue is read to establish the current position. The updated entry is rewritten to the tracking queue and the routine to build the panel image is performed.

If this update action is taking place via a LIST panel, the LISTtrmid TS queue is read to obtain the account number of the record to be updated and DFH0VRIO is called to read the customer file. The LISTtrmid TS queue is rewritten with a code to show that the record is held for update. If this panel is not being processed via a LIST panel, it is not necessary to call DFH0VRIO, because the DFH0VOPN module will have obtained the single data record. The RECDtrmid TS queue entry is rewritten with a code to show that the record is held for update.

The data is moved from the TS queue layout into the map fields. The appropriate panel identifier is set up to control the function key detail line. If all the records selected for processing via the LIST panel have been processed then the function key line without F5 is required, otherwise the function key line that includes F5 must be displayed. The function key line table is searched and the correct line is moved into the panel output field. This completes building the panel, which is now written to the PANLtrmid TS queue. The cursor is positioned on the main field (surname), the panel is sent and control is returned to CICS with AC26 as the next tranid.

When F10

The PANLtrmid TS queue entry is read, the cursor is positioned on the main panel field (surname), the panel is sent, and control is returned to CICS with AC26 as the next tranid.

When F12

The first task is to rebuild the base panel image. This is done by reading the stored image queue (PANLtrmid TS queue) for the current entry. Cursor

positioning on the panel may be either symbolic or specific. If a return from F1 is in progress then the stored cursor position (EIBCPOSN), is used to give the specific numeric cursor position. Any other cursor positioning is entirely symbolic and dependent on the COMMAREA parameters. If a pull-down has yet to be displayed over the base map to complete the rebuilt image then no cursor positioning is performed.

Once the panel image has been rebuilt it is sent to the screen and then a test is performed to see whether this base panel image completes the rebuilding of the screen. If processing is complete control is returned to CICS with AC26 as the next tranid; otherwise the tracking entry of the RECDtrmid TS queue is read and used to route to the next program via DFH0VTBL. Specific processing is built in here to allow for control having to be passed to the DFH0VOL program to display a pull-down. This routing is done via the DFH0VTBL program rather than directly so as to maintain consistency.

When CLEAR

The CLEAR key is processed in the same way as F12, except that there is no need to consider routing to another module if a pull-down needs to be displayed over the top of this panel. The reason for this is that the 'reset panel' code will be executed when processing CLEAR from the pull-down module.

'Reset panel' processing

This is invoked by the pull-down module, DFH0VOL, when the user presses CLEAR, moves along the action bar or selects an option that results in a pop-up.

The 'reset panel' code reads the PANLtrmid TS queue to find the current panel image, which is then sent to the screen before control is passed either back to the DFH0VOL module or to the next required panel as necessary.

There are two special cases dealt with in this processing, to allow for the user selecting option 8 (exit) or option 4 (save) from the FILE pull-down. The exit processing is the same as described for F3. The save processing reads the PANLtrmid TS queue and rebuilds the panel image. If the record was not successfully validated before the attempt to save it, a message is returned; otherwise the indicators are tested to see whether this update is being processed as an individual record or via a LIST. The appropriate TS queue is updated to show that the update is complete, the panel image is sent, and control is returned to CICS with AC26 as the next tranid.

Action on invocation by transaction code AC26

After a return to CICS with AC26 as the next tranid, the program is reentered when the user presses a function key. First the PANLtrmid TS queue is read to establish the current position, then the map is received. After this the function key area is built and the updated panel image rewritten so that if the user selects a pull-down from the action bar, previous updates will not be lost. They are not added to the file because they have not been validated but the user can still be restored to his previous screen position.

When F1

The field cursor attributes are tested to determine on which field contextual help is required. The appropriate commarea parameter is set up and passed via the routing program to DFH0VHP. Before control is transferred, the current cursor position (EIBCPOSN) is saved in the commarea to allow for later repositioning of the cursor at the field from which help was requested.

When F3

Processing is the same as when entering the program via a transfer of control.

When F5

If the update is being processed via a LIST but the end of the list has been reached, F5 is invalid and an appropriate message is returned. Otherwise control is transferred via the DFH0VTBL program to the DFH0VLST program to allow it to pass processing on to the next module required from the selection LIST.

When F10

The only processing required is to determine the new position of the cursor. If it is on the action bar, it is moved to the main panel field (surname) and if it is within the UPDATE panel, it is moved to the first action bar field, FILE. The cursor is resent via a 'control send', and control is returned to CICS with AC26 as the next tranid.

When F12

The same processing as described for F3 is performed to free the held record entry and find the previous LISTtrmid TS queue record entry if necessary. Then the tracking queue is read, the entry for the current panel is cleared and the previous base panel is found. Control is transferred to the required base panel routine to allow the rebuilding of the image to begin.

When CLEAR

The required processing is to rebuild the panel from the PANLtrmid TS queue, reposition the cursor as described for F12 via a transfer of control, send the panel, and return control to CICS with AC26 as the next tranid.

When ENTER

If the cursor is positioned on the action bar to request the display of a pull-down, an indicator is set in the required COMMAREA fields and control is transferred to the pull-down display module DFH0VOL via the routing program.

If the user presses ENTER when the cursor is not on the action bar, nominal validation is carried out on the data on the screen. The surname is checked to see that it is alphabetic and then an indicator is set to show that the record has been validated. If the record is found to be valid the RECDtrmid TS queue entry is updated with the data from the screen; otherwise an appropriate error message is sent with an alarm.

Program DFH0VDEL – delete customer details panel processing

This program is invoked by transaction AC27 and performs the processing necessary when a user wants to delete a particular record or when a user selects the delete action from a selection list.

It is initiated via a transfer of control either from the pull-down module (DFH0VOL) or from the list panel module (DFH0VLST). After presenting the initial panel all further interaction is performed via the AC27 transaction.

The valid function keys within this panel are:

F1 Request contextual help.

F3 Return to the T1 base panel.

F5 Allow forward processing through a selection list from a list panel.

F10 Toggle the cursor between the action bar and the main cursor position (account number).

F12 To return to the previous panel.

Action on invocation by transfer of control

When entering via a transfer of control, processing of the function keys is as follows:

When F3

The transaction-related resources are cleared up and processing is performed to route to the previous panel in the tracking queue (item one of the RECDtrmid TS queue). If the previous panel was a pop-up, the DFH0VTBL routine is called at the action and base levels to establish the current position; otherwise control is just transferred to the previous module. Clearing up the transaction-related resources involves freeing any records that are held on the customer file for delete processing but which have not been completely deleted. This processing is the same as for F12.

If the delete is being processed via a LIST panel, the LISTtrmid TS queue is read to obtain the record key, the remote I/O module (DFH0VRIO) is called to make the appropriate record available and the LISTtrmid TS queue is searched to find the next entry for processing. If the delete is not being processed via a LIST panel, the RECDtrmid TS queue is read to check the current state of the record (if the delete is complete no freeing is necessary) and then the DFH0VRIO module is called to perform the appropriate actions on the record.

When F5 or ENTER

The tracking queue (item 1 of the RECDtrmid TS queue) is updated with the delete panel details. Then the panel TS queue is read to establish the current position.

The updated entry is rewritten to the tracking queue. If this delete action is taking place via a LIST panel the panel-building routine is performed. The LISTtrmid TS queue is read to find the account number of the record and DFH0VRIO is called to read the customer file. The LISTtrmid TS queue is rewritten with an indicator to show that the record is held for deletion.

If only a single record is being processed, without going via a LIST panel, the RECDtrmid TS queue entry is rewritten with an indicator to show that it is held for deletion.

The record is read from the RECDtrmid TS queue and the data is moved from the TS queue layout into the map fields. The appropriate panel identifier is set up to control the function key line. If all the items selected from the LIST panel have been processed, the flag for the function key line without F5 is set; otherwise the flag for the function key line including F5 is used. The required function key line is then moved into the panel output field.

The fields on the delete panel are all protected except for ‘account number’, which is used initially to select the record when processing from the pull-down option. Once the record is found, a confirmation indicator is set to show that the next ENTER key is to confirm the deletion and therefore the account number is protected.

This completes the panel building process and the panel is now written to the PANLtrmid TS queue. The cursor is moved to the account number field, the panel is sent, and control is returned to CICS with AC27 as the next transid.

When F10

The panel TS queue entry is read, the cursor is set on the main panel field (account number), the panel is sent, and control is returned to CICS with AC27 as the next tranid.

When F12

If F12 is processed via a transfer of control then the first task is to rebuild the base panel image. This is done by reading the stored image queue (PANLtrmid) for the current entry.

Cursor positioning on this panel may be either symbolic or specific. If a return from an F1 is in process the stored EIBCPOSN field is used to give the specific numeric cursor position. Any other cursor positioning is entirely symbolic and dependent upon the commarea parameters. If a pull-down has yet to be displayed over this base map no cursor positioning is performed.

Once the cursor is in position, the panel image is sent and a test is performed to see if this base panel image completes the rebuilding or if routing must be performed to another module. If processing is complete control is returned to CICS with AC27 as the next tranid; otherwise the tracking queue is read and used to route to the next program via DFH0VTBL. Specific processing is built in here to allow for a pull-down being required and control having to be passed to the DFH0VOL program. This is done via the DFH0VTBL program, rather than directly, to maintain consistency.

When CLEAR

The CLEAR key functions in the same way as F12 except that there is no need to consider routing to the next module if a pull-down needs to be displayed over the top of this panel. The reason for this is that processing CLEAR from the pull-down module executes the 'reset panel' code.

'Reset panel' processing

This is invoked by DFH0VOL, the pull-down module, when the user presses CLEAR, moves the cursor along the action bar, or selects an option that results in a pop-up.

The 'reset panel' code reads the PANLtrmid TS queue to find the current panel, which is then sent to the screen before control is passed either back to the DFH0VOL module or to the next required panel as necessary.

A special case is dealt with in this process, to allow for the user selecting option 8 to exit from the FILE pull-down. This exit processing is the same as for F3 described above.

Action on invocation by transaction code AC27

After a return to CICS with AC27 as the next tranid, the program is reentered when the user presses a function key. The PANLtrmid TS queue is read first in order to establish the current position, and then the map is received. The valid function keys are processed as follows.

When F1

The field cursor attributes are tested to determine on which field the user requires contextual help. The appropriate commarea parameter is set up and passed via the table searching program DFH0VTBL to the help program

DFH0VHP. Before control is transferred the current cursor position is saved in the commarea to allow for later repositioning on the correct field.

When F3

This processing is the same as when control is transferred from another transaction.

When F5

The user may press F5 to scroll through a list of items. Control is transferred, via the routing program DFH0VTBL, to the DFH0VLST program to allow the next panel to be selected for display.

When F10

The only processing required is to determine the new position of the cursor. If it is currently displayed on the action bar it is moved to the account number field or if it is already on the account number field it is moved to FILE, the first action bar field. The newly positioned cursor is sent via a control send and control is returned to CICS with AC27 as the next tranid.

When F12

The same processing as for F3 is performed to free the held record entry and find the previous record entry in the queue LISTtrmid. Then the tracking queue is read, the entry for the current panel is cleared, and the previous base panel is found. Control is transferred to the required base panel module to allow the rebuilding of the image to begin.

When CLEAR

The panel is rebuilt from the PANLtrmid TS queue. The cursor is repositioned as described for the F12 transfer of control processing, the panel is sent, and control is retuned to CICS with AC27 as the next tranid.

When ENTER

When the user presses ENTER with the cursor positioned on the action bar to request the display of a pull-down, the cursor position is detected and indicators are set in the commarea. Control is transferred to the pull-down display module DFH0VOL via the table router.

If the user presses ENTER when the ‘confirm deletion’ indicator is set, the DFH0VRIO module is called to actually delete the record from the file. The appropriate TS queue, either LISTtrmid or RECDtrmid, is updated to indicate that the deletion is complete. The function key area is rebuilt, the map is resent, and control is returned to CICS with AC27 as the next tranid. The confirmation indicator is turned off.

If the user presses ENTER when the confirmation indicator is turned off, an attempt is made to perform the processing described for F5 and ENTER via a transfer of control from a list. The panel building routine is performed, the confirmation flag is set, the function key area is set up, the panel is sent, and the AC27 transaction code is returned to CICS.

Program DFH0VOL – overlay handler

This program is invoked by transaction AC21 and performs the processing required when a selection is made from the action bar.

The base panel that is currently displayed, transfers control to this module indicating which pull-down has been selected and so which processing should occur. When the pull-down has been displayed the user can select the available options from it. The selections are validated and control transferred dependent upon the type of panel to be displayed, for example a base panel and a pop-up, a 'full screen' pop-up, or a pull-down and a pop-up.

The valid function keys within this panel are

F1 Request contextual help.

F3 Return to the T1 base panel, or to exit to CICS if the pull-down is displayed over the T1 panel.

F10 Remove the pull-down, returning the cursor to the main cursor position on the base panel.

F12 Return to the previous panel, returning the cursor to the action bar.

Action on invocation by transfer of control

The function keys are processed as follows:

When F12

Processing depends upon which pull-down is displayed and whether the display of the pull-down completes the image. If the image is complete, control is returned to CICS with AC21 as the next tranid. If not, the pull-downs are rebuilt and sent, and control is transferred to the next appropriate module via the routing program, DFH0VTBL.

When CLEAR

Processing is exactly the same as for F12.

When ENTER

The requested pull-down panel is built and displayed. The options available in the pull-down can vary according to the current base panel. Before the panel is sent, the options are passed to the DFH0VTBL module which checks what options are available according to the current action and base level. The options that are not available are displayed in blue with an asterisk '*' replacing the option number. When all the options have been checked, the panel is sent and control is returned to CICS with AC21 as the next tranid.

Action on invocation by transaction code AC21

The appropriate pull-down map is received and the valid function keys are processed as follows:

When F1

The field cursor attributes are tested to determine on which field contextual help is required. The appropriate commarea parameter is set up and passed via the routing program to DFH0VHP.

When F3

The tracking queue entry is read to find the previous entry to which to transfer control. According to the rules of CUA this must be a base panel, because it is possible to select a pull-down only from an action bar on a base panel.

When F10

The processing is the same as described for F3.

When F12

The tracking queue (item one of the RECDtrmid) is read to find the base panel that is under the current pull-down. Commarea parameters are set so that the panel rebuild will stop after the base panel. Control is transferred to the base panel program via the routing program DFH0VTBL.

When CLEAR

Processing is the same as described for F12 except that the pull-down is redisplayed as well as the base panel.

When ENTER

The actual processing of the user's selections from the pull-downs takes place. The processing varies depending on the pull-down but follows a pattern of general validation, specific validation, and action.

FILE pull-down processing. The first action is to check if the cursor is positioned on the HELP action bar field. If it is, the appropriate parameters are set up to force the base panel to be reset and the HELP panel to be displayed over the top.

The selection entry field is checked to see that it is not alphabetic, and that it is a valid option. If it is not valid the panel is redisplayed and control is returned to CICS with AC21 as the next tranid.

If the selection is '8' then an indicator is set to trigger 'exit' processing throughout the application.

If one of the currently active panels (in the tracking queue) is a list panel and any selection other than SAVE or SAVE AS is made, the list indicator in the commarea is set to show that this list is being displayed, not processed, and the tracking queue is set back to point to the list panel as the current base panel. This is to allow for the termination of action list processing via the pull-down, whilst still allowing SAVE and SAVE AS to perform correctly.

If the SAVE option is selected (option 4) then the DFH0VRIO module is called to save the record from the RECD TS queue to the CUSTOMER file. This is done only if the record has been successfully validated (an indicator is set in the COMMAREA).

If the SAVE AS option is selected (option 5) and the record has not been validated then control is transferred to the base panel via the DFH0VTBL routine after reading the index data from the tracking queue.

The final stage in processing the FILE pull-down is to transfer control to the next module that will build the selected panel image. This is dependent upon the panel type. If the option selected leads to a base panel then control can be transferred straight there. The program name is already known but position in the action/base table must be established, using the last three characters of the program name as a key. For a FULL SCREEN pop-up it is only necessary to transfer control straight to the processing module. For a partial pop-up that results from a selection, the base panel must be redisplayed first. A base level

search is performed through DFH0VTBL to establish the position and control is transferred to the base processing module with a status code of 'RESET' to force it to reset the panel.

HELP pull-down processing. The HELP pull-down processing is almost the same as described for the FILE pull-down except for the option-specific validation. For the HELP pull-down the first check is to see if the cursor is positioned on the FILE action bar field to select the FILE pull-down for presentation. If it is then the appropriate parameters are set up to force the base panel to be reset and the FILE panel to be displayed over the top. The selection entry field is checked to see that it is not alphabetic. The selection is then checked to see if it is unavailable or invalid. After this, control is transferred to the next required module in the same way as described for FILE pull-down processing.

Program DFH0VOPN – open file pop-up handler

This program is invoked by transaction AC22. It performs the processing required when the 'open for browse' or 'open for update' options are selected from the FILE pull-down, or when the 'open for browse' pop-up is obtained by the *fastpath* route of pressing ENTER from the T1 panel.

The OPEN pop-up is initially displayed as a result of a transfer of control from the currently active base panel and is subsequently redisplayed when transaction AC22 is invoked by CICS.

The only valid function keys within this panel are:

F1 Request contextual help.

F12 Return to the previous panel.

CLEAR is an invalid function key but is handled specifically by this application to enable the rebuilding of the current panel image(s).

Action on invocation by transfer of control

The function keys are processed as follows.

When F3

The transaction-related resources are cleared from the TS queues and processing is performed to route to the previous panel in the tracking queue. Depending on the system design, the previous panel must be a base panel; however, there is a test to see what sort of panel it was as an example of how to process a pop-up resulting from a previous pop-up.

When F12

The first step is to read the current panel TS queue entry and send the OPEN panel image. A test is performed to see if sending this panel completes the required image. If the panel building is complete, control is returned to CICS with AC22 as the next tranid; otherwise the tracking queue is read for the next entry and control is routed to the appropriate program for it to present the required image.

When CLEAR

Processing is the same as described for F12.

When ENTER

An entry is written to the tracking and panel TS queues and an OPEN panel is built with all the entry fields filled with blanks to block out any underlying display. The panel is sent and control is returned to CICS with AC22 as the next tranid.

Action on invocation by transaction code AC22

The panel queue is read to find the current item number so that the panel can be rewritten. The map is received and then processing continues, depending on which function key was pressed.

When F1

The field cursor attributes are tested to determine on which field contextual help is required. The appropriate commarea parameter is set up and is passed via the routing program DFH0VTBL to the DFH0VHP program.

When F12

The current entry (the one with the OPEN panel in it) is cleared from the TS queues and the tracking queue record (item 1 of the RECDtrmid TS queue) is searched to find the previous base panel. The tracking queue is updated and DFH0VTBL is called to determine the program to which control should be transferred in order to start rebuilding the panel.

When CLEAR

The same process as for F12 is performed, except that the current entry is not cleared.

When ENTER

The actual panel processing takes place. This involves verifying the key fields entered and if they are valid, calling the File I/O module (DFH0VRIO) to access the customer file and read the appropriate records. If the keys are not valid, fields in error are highlighted in yellow and reverse video. On returning from the I/O module it is determined whether the keys supplied identified a single record or a list of records. To process a list, control is transferred to DFH0VLST; otherwise control is transferred to DFH0VBRW or DFH0VUPD depending on which FILE pull-down option was selected.

Note that the programs are not specified directly by their names but are called dynamically by the program DFH0VTBL.

Program DFH0VPRT – print pop-up handler

This program is invoked by transaction AC28 and performs the processing required when the PRINT pop-up is selected, that is, option 7 from the FILE pull-down.

It is initiated as a result of a transfer of control from a base panel and is subsequently reentered when transaction AC28 is invoked by CICS.

The only valid function key within this panel is:

F12 Return to the previous panel.

The map is not received since no data can ever be entered on this panel and contextual help is not available on any of the fields.

The CLEAR key is invalid under CUA rules but this application handles it by refreshing the screen.

Action on invocation by transfer of control

Processing is dependent upon which function key has been pressed.

When ENTER

The panel is built and an entry written to the tracking queue (item one of the RECDtrmid TS queue). The screen image is written to the PANLtrmid TS queue, the panel is sent, and control is returned to CICS with AC28 as the next tranid.

When CLEAR

The panel TS queue is read, the image is sent, and control is returned to CICS with AC28 as the next tranid.

Action on invocation by transaction code AC28

When F12

The tracking queue is read, the current entry (the one with the PRINT panel in it) is cleared and the previous base panel is located. The DFH0VTBL module is called to determine the program to which control should be transferred in order to start rebuilding the panel.

When CLEAR

The same action as for F12 is performed, except that the current entry is not cleared. Once the application transfers control back to this module, having rebuilt all of the previous images, the PANLtrmid TS queue is read, the panel is sent, and control is returned to CICS with AC28 as the next tranid.

Program DFH0VSAS – save customer details pop-up handler

This program is invoked by transaction AC2A and performs the processing required when the SAVE AS option is selected, that is, option 5 from the FILE pull-down.

The panel is initially displayed as a result of a transfer of control from the currently active base panel and is subsequently reentered when transaction AC2A is invoked by CICS.

The only valid function keys within this panel are:

F1 Request contextual help.

F12 Return to the previous panel.

CLEAR is an invalid function key under the CUA rules, but is handled specifically by this application to refresh the screen.

Action on invocation by transfer of control

Processing is dependent upon which function key has been pressed.

When F3

The transaction-related resources are cleared and processing is performed to route to the previous panel in the tracking queue. The code contains a test to see if the previous entry was a pop-up or any other type of panel, although the system design requires that it should always be a base panel. This code is provided as an example of how to process a pop-up that results from a previous pop-up.

When F12

The current PANLtrmid TS queue entry is read and the SAVE AS panel image is sent. A test is performed to see if this panel completes the required image, and if so, control is returned to CICS with AC2A as the next tranid; otherwise the tracking queue is read to find the next entry to which control should pass.

When CLEAR

Processing is exactly the same as described for processing F12.

When ENTER

An entry is written to the tracking and panel TS queues, and the DFH0VRIO routine is called to allocate the next available record on the file. The SAVE AS panel is sent, showing the record to be allocated and requesting confirmation from the user. Control is then returned to CICS with AC2A as the next tranid.

Action on invocation by transaction code AC2A

After a return to CICS with AC2A as the next tranid, the program is reentered when the user presses a function key. The SAVE AS panel is received into the data area and the valid function keys are processed as follows.

When F1

The field cursor attributes are tested to determine on which field contextual help is required. The appropriate commarea parameter is set up and passed via the routing program to the DFH0VHP program.

When F12

The tracking queue is read. If the record has been allocated the I/O routine, DFH0VRIO, is called to delete it so that it is not left in a partially allocated state on the file. The current entry (the one with the SAVE AS panel in it) is cleared and the tracking queue (ITEM 1 of the RECDtrmid TS queue) is searched for the previous base panel. After the tracking queue has been updated the DFH0VTBL module is called to determine the program to which control should be transferred to start rebuilding the panel image.

When CLEAR

Similar processing to that described for F12 is performed except that the current entry is not is not cleared and the partially allocated record is not deleted.

When ENTER

The actual saving of the data takes place. This involves calling the I/O module (DFH0VRIO) to save the data in the newly allocated record. Control is transferred to the previous base panel by reading the previous track entry and routing via the program (DFH0VTBL). This method of implementation allows this pop-up to be made available over any base panel.

Program DFH0VHLP – help pop-up handler

This program is invoked by transaction AC2C and performs the processing required when the HELP full screen pop-up is displayed, that is, when the user selects one of the options 1 through 5 from the HELP pull-down.

It is initiated as a result of a transfer of control from the overlay module (DFH0VOL), and is subsequently reentered when transaction AC2C is invoked by CICS.

The only valid function key within this panel is:

F12 Return to the previous panel.

The map is not received since no data can ever be entered on this panel and contextual help is not available on any of the fields.

The CLEAR key is invalid under CUA rules but this application handles it by refreshing the screen.

Action on invocation by transfer of control

When ENTER

The panel is built and an entry is written to the tracking queue (item one of the RECDtrmid TS queue). The screen image is written to the PANLtrmid TS queue, the panel is sent, and control is returned to CICS with AC2C as the next tranid.

Action on invocation by transaction code AC2C

When F12

The tracking TS queue is read, the current entry (the one for the HELP pop-up) is cleared, and the previous base panel is located. The DFH0VTBL module is called to determine the program to which control should be transferred in order to start rebuilding the panel from the previous base panel.

When CLEAR

The panel is redisplayed from the current PANLtrmid TS queue entry and control is returned to CICS with AC2C as the next tranid. This simple processing is possible because no panels can be overlaid on this full screen pop-up.

Program DFH0VHP – contextual help pop-up handler

This program is invoked by transaction AC2E and performs the processing required when the CONTEXTUAL HELP pop-up is built as a result of the function key F1 being pressed while the cursor is on a valid field.

The panel is initially displayed as a result of a transfer of control from the current panel and is reentered when CICS invokes the AC2E transaction. This panel can be on the screen at the same time as a pull-down but no panel can be displayed over the top of it.

The only valid function key is:

F12 Return to the previous panel.

Action on invocation by transfer of control

When F1

Processing starts by writing an entry to the tracking queue to show that this is the current panel. The local I/O processor, DFH0VLIO, is called to access the contextual help text for the required field. The panel is built from the HELPtrmid TS queue built in the DFH0VLIO module. When the panel is ready, it is sent to the screen and control is returned to CICS with AC2E as the next tranid. The actual screen image is written to the PANLtrmid TS queue.

When CLEAR

The current entry from the PANLtrmid TS queue is read, the image is sent to the screen and control is returned to CICS with AC2E as the next tranid. The generated BMS DSECT is redefined by a user defined DSECT, which allows the use of GROUP fields and ARRAYS to enable much more compact and modular processing. Care must be taken to ensure that if ever the main BMS DSECT is changed. the user-defined DSECT is kept in line with it. Failure to do so can lead to PROG402 errors, data checks, and so on.

Action on invocation by transaction code AC2E

When the transaction AC2E is invoked by CICS the map is not received as there are no input fields on it. The valid function keys are processed as follows.

When F12

The HELPtrmid TS queue, which holds the current help information, is deleted. The tracking queue is read and the latest entry (for the HELP pop-up) is cleared. The previous base is found and control is transferred to the appropriate module via the DFH0VTBL routing program.

When CLEAR

The processing is the same as for F12 except that the HELPtrmid TS queue is not deleted and the current tracking entry is not cleared.

Other

All other function keys are invalid. The panel is sent with an error message and control is returned to CICS with AC2E as the next tranid.

Program DFH0VABT – about pop-up handler

This program is invoked by transaction AC2F and performs the processing required when the ABOUT pop-up is selected, that is, option 6 in the HELP pull-down.

It is initiated as a result of a transfer of control from a base panel, and is subsequently reentered when transaction AC2F is invoked by CICS.

The only valid function key within this panel is

F12 return to the previous panel.

The map is not received since no data can ever be entered on this panel and contextual help is not available on any of the fields.

The CLEAR key is invalid under CUA rules but this application handles it by refreshing the screen.

Action on invocation by transfer of control

When ENTER

The panel is built and an entry is written to the tracking queue (item one of the RECDtrmid TS queue). The screen image is written to the PANLtrmid TS queue, the panel is sent, and control is returned to CICS with AC2F as the next tranid.

Action on invocation by transaction code AC2F

When F12

The tracking TS queue is read, the current entry, (the one for the ABOUT pop-up), is cleared and the previous base panel is located. The DFH0VTBL module is called to determine the program to which control should be transferred in order to start rebuilding the panel from the previous base panel.

When CLEAR

The panel is redisplayed from the current PANLtrmid TS queue entry and control is returned to CICS with AC2F as the next tranid. This simple processing is possible because no panels can be overlaid on this full screen pop-up.

Program DFH0VTBL – table router

This program is not invoked by a transaction. It is called dynamically by the other modules in the application.

It matches parameters in the commarea against prebuilt user tables in order to find the names of the programs to which control may be passed. This table searching saves the programmer of each module having to know which options are available at each particular point in the application. The table maintains this information, and also gives an amount of flexibility in program naming, testing and installing since test versions of programs could easily be substituted for production source simply by changing the table entry and recompiling DFH0VTBL.

The table has three dimensions, action, base, and selection. The CUA text model application has only one object, the customer data file, but if there were more than one object, this would be a four-dimensional table with the object as the highest level.

There are six types of search that can be called for:

1. Action level
2. Base level
3. Selection level
4. Action and base level
5. Action and base to DFH0VOL
6. Function key search.

Searches at the different levels are processed as follows.

Action level

The ‘action selected’ field in the commarea is matched against the table. If a match is found, the entry is positioned at this point in the table. If no match is found, the action is invalid.

Base level

The ‘base indicator’ from the commarea is matched against the table. If a match is found, the matching entry contains the program name which is returned to the calling program. If no match is found, the base panel is flagged as not available.

Selection level

The 'selection' field from the COMMAREA is matched against the table. If a match is found, the matching entry contains the program name and program type, which are passed back to the calling module.

There are various switches and indicators that can be set during application processing, therefore, further testing must be performed. When any selection program name in the table is set to 'TURNDOFF' it has been turned off by the application designer, because it is not available at this point in the application. For example, if the LIST panel has been processed, it is not possible to select options 2 or 3 (OPEN FOR BROWSE or UPDATE) because it is not possible to process multiple lists.

If the selection is not found, the processing goes on to check the function key table. If the selection is found, the program name and type are returned; otherwise the invalid selection indicator is set and the program name DFH0VOL is returned.

Action and base levels

The action and base level searches are performed as described above and the resulting program name is returned.

Action and base levels returning to the DFH0VOL program

This type of search is used when a pull-down is requested from a base panel. The index indicators are positioned on the action and base dimensions and DFH0VOL is set as the name of the program to return to.

Function key table search

The function key table is a separate table that is also accessed by this module. It maintains a list of program names by function key, and is used to find the programs that can be initiated by the user pressing a function key, for example, F1 for DFH0VHP, the contextual help program. It is searched using alphabetic keys that do not occur on the selection table. If the key is found the program name and type are returned; otherwise the invalid key indicator is set and the program name DFH0VOL is returned.

Program DFH0VAB – abend handler

This program is invoked by transaction AC2D and is called from any application module that terminates with a CICS abend.

It presents a panel showing the name of the module that abended, the number of the CICS call that was last executed, the actual abend number from EIBRESP and the resource that was being operated upon when the abend occurred, (EIBRSRCE).

The only valid function key within this panel is:

F3 Erase the panel and return control to CICS.

Program DFH0VRIO – customer data file handler

This program is not invoked by a transaction, it is linked to by the other programs in the application.

This is the remote I/O handling module. Its purpose is to process the I/O functions on the customer file (DFH0FCUS) and the customer file secondary index (DFH0FCAI). The functions performed include accessing data from the file and updating records on it. These operations are driven by parameters in the COMMAREA and data is passed between the calling module and this operating module via the RECD and LIST TS queues. The module is designed such that other files could be added to it with only minor modifications. The intention is that eventually this module could be installed on a Data Owning Region (DOR) that is accessed across an APPC link. The commarea and any required data (for example, updated records) could be passed across the link to this module for processing and the appropriate response passed back. This implementation would require several new modules to perform the APPC processing, and slight modification to any front-end modules that call DFH0VRIO, to make them call the new APPC front end processor.

The processing in the module is based on I/O processing type. There are seven types:

1. Read Record
2. Read for Update
3. Free Record
4. Allocate Record
5. Write Record
6. Delete Record
7. Save Record.

They function as follows.

Read Record ('RR')

This can operate on either the primary index, 'customer number', or the secondary index, 'customer name'.

If data has been entered in the 'customer name' field of the OPEN panel (OPN), the secondary index is processed. There are three different ways of reading the secondary index,

1. With a specific key so that all records with that key are read
2. Generically, using a partial key followed by an asterisk (*)
3. Using an asterisk (*) as a wild card so that the whole file is read.

These three methods all result in the same basic processing. A STARTBR is performed. If the response is normal, the record is read; if not, the condition is handled in the calling module. The entry is then written to the list TS queue and further records are read and copied to the TS queue until one is found whose key does not match the key specified by the user.

If a Read Record is requested and the customer field is blank but a customer number (or range of numbers) has been entered, similar processing is performed on the primary index. A STARTBR is followed by reading the

records with the specified customer number(s), and the entries are written to the list TS queue. If the Commarea indicators show that a record is to be updated, the Read for Update processing is performed as described below.

If a Read Record is requested with any other status, a single read on the primary index is performed. The record is written to the RECDtrmid TS queue, and if the LIST panel is being processed the tracking queue is updated, otherwise the tracking queue is updated in the calling module.

Read for Update ('RU')
Processing is performed on the primary index. The RECD TS queue is read, and then the customer file is read for update. A check is made to see whether the record is in use by another user. There is a 4-byte indicator field on the end of each record to identify any user who is holding a record for update. If the record is already in use, a return code is set to indicate the fact to the calling program. If the record is available the user's terminal ID is put in the indicator to show that the user has it for update, and the record is written to the RECD TS queue.

Free Record ('FR')
This processing is used when records are held for delete or update but the user has chosen to exit from this processing without completing it. The customer file is read with the account number as the key, and the record locking indicator is reset.

Allocate Record ('AL')
This process is called from the NEW panel to allocate the first available account number on the file for a new record to be created. The customer file is scanned to find the lowest available customer account number and the record found is marked as locked by the user. The record entry is written to the RECDtrmid TS queue.

Write Record ('WR')
This process simply reads the record from the RECD TS queue and writes it to the customer file using the primary index.

Delete Record ('DD')
This process is performed from the DELETE panel to delete a specific record from the customer file, and is also used from the NEW panel to delete any dummy entries that are not required because the user has chosen to abandon processing. The RECD TS queue is read to obtain the key of the record to be deleted and then the delete is performed against the customer file using the primary index. If any errors occur they are dealt with in the calling module.

Save Record ('SR')
This process performs the Read for Update procedure to transfer the record to the RECD TS queue. The user's updates are applied to the record, which is then re-written to the customer file.

After the required type of I/O processing, the LISTtrmid TS queue is examined and if there is only a single entry in it, it is turned into a RECDtrmid TS queue. This is because the design dictates that an inquiry that results in only one record being read should transfer control to the appropriate base panel, not the list panel.

Control is then returned to the calling program.

Program DFH0VLIO – help file handler

This program is not invoked by a transaction, but is linked to by the module DFH0VHP.

This is the local I/O handling module. Its only purpose is to read the local HELP file (DFH0FHLP) and create the HELPtrmid TS queue from which the DFH0VHP module can build the 'help' pop-ups. It could be adapted to process any other local files as it is parameter driven from the commarea and the resulting data is always returned via temporary storage.

The only I/O request type passed to this module is 'RR' - Read Record. This causes a key field to be built, consisting of a screen field name, for example ACCNO for account number, and a line counter. There can be up to thirteen lines of help information for each field, including the title. A routine is performed, which uses this key field to read the lines of help information from the file and copy them to the HELPtrmid TS queue.

If the user presses F1 while the cursor is not on a valid field for contextual help, the key field is set to spaces. There is a special entry in the file with a key field of spaces, which explains to the user how to update and access the help file.

When all the required lines of help information have been read this module returns control to DFH0VHP.

Chapter 10. CUA text model file and copybook descriptions

This chapter gives a brief description of the various files and copybooks used in the CUA text model application programs.

File: DFH0FUS customer detail file

This is a VSAM KSDS consisting of records 227 bytes long with an 8-byte key which is the customer account number. Each record maintains the details for a particular customer, for example name and address.

File: DFH0FAI customer detail file - alternate index

This secondary index file points to DFH0FCUS, the customer detail file. Its key field is the customer surname.

File: DFH0FHLP help pop-up data file

This is a VSAM KSDS consisting of records 38 bytes long with a key in positions 1 to 10.

The records in this file are used to build the text for a contextual help pop-up. The key is made up of the field name from which contextual help was requested and a line number. There can be up to 13 lines of text on each help pop-up. The field name keys should match the field names in the DFH0BHT copybook. If the cursor position does not correspond to one of these field names, text with blank field name is retrieved from the file, explaining that no help information is available on the field selected.

Copybook: DFH0BCR customer record layout

This copybook shows the detail fields that make up the customer records on the DFH0FUS file. The field names are self-explanatory. The USERTERMID field is used when an update function requires exclusive control of a record for a particular terminal. This is one of several possible methods which can be used to maintain update integrity.

The physical record is 227 bytes long but the layout is 228 bytes long. This is because the copybook is used to write records to the RECDtrmid TS queue and the last byte is used to identify the operation being performed on the record, and its current status.

Copybook: DFH0BCA commarea

The commarea layout is the same across all modules. It is 200 bytes long with 61 bytes of filler at the end. The fields are detailed in Chapter 12, “Installing and running the FILEA sample applications” on page 121. Included in the copybook are fields that are used in most of the modules within the application, including the required literals.

© Copyright IBM Corp. 1989,1999

Copybook: DFH0BFKT variable function key layout

This is an array of literals that are used to build the function key line for panels that have different function keys available at different times. The panel identification field is the key to this array.

Copybook: DFH0BFPD redefinition of file pull-down DSECT

This redefinition is necessary because the keywords GROUP and OCCURS are incompatible when using BMS. The redefinition does not use any extra storage but enables easier code manipulation by grouping similar fields.

It is important to note that any changes made to the file pull-down DSECT must be propagated into this copybook to prevent DATACHECKs from occurring.

Copybook: DFH0BHPD redefinition of help pull-down DSECT

This redefinition is necessary because the keywords GROUP and OCCURS are incompatible when using BMS. The redefinition does not use any extra storage but enables easier code manipulation by grouping similar fields.

It is important to note that any changes made to the help pull-down DSECT must be propagated into this copybook to prevent DATACHECKs from occurring.

Copybook: DFH0BHP redefinition of help pop-up

This is a redefinition of the contextual help pop-up DSECT. It is an array of 13 lines that are built from the DFH0FHLP file.

Copybook: DFH0BHT help file key table

This table contains the keywords that relate the cursor-detected fields to the text in the DFH0FHLP file.

Copybook: DFH0BLST redefinition of the list base panel

This DSECT redefinition is used to build the customer detail lines on the list panel.

Copybook: DFH0BMSG application message table

This table contains all the error and warning messages used by the modules within this application.

Copybook: DFH0BRT program routing control table

This is a 3-dimensional table that controls the flow of the application from module to module. The three dimensions are:

1. Actions
2. Base Panel
3. Pull-down options.

Each action on the action bar has an entry in the table, within which there is an entry for every base panel to identify the associated program for processing. Within each base panel, the pull-down options show which program is to be used to process the option and whether it results in a base panel (B), a pop-up (P), or a full screen pop-up (F). If the program name is TURNDOFF, that particular program is not available from the current base panel.

This table allows for new programs to be added and tested without changing any code within the existing modules. It eliminates decision making from the application as far as program routing is concerned.

If multiple objects were available, a fourth dimension called *object* could be added to the table.

Copybook: DFH0BTSQ TS queue details

This copybook shows the layout of the items in the LISTtrmid TS queue and the tracking entry in the RECDtrmid TS queue. The limit of 50 occurrences in the tracking record is an application choice.

This list TS queue holds the details necessary to produce. the list panel selection fields. It also holds details of the type and status of processing being performed against each entry.

Copybook: DFH0BHR help text TS queue layout

This TS queue layout is used to pass the help data from the DFH0VLIO module to the DFH0VHP module that displays the contextual help pop-up.

Chapter 11. CUA text model BMS maps

All the maps used in the CUA text model application are coded for minimum function BMS, and also the CURSLOC function. To run this application on earlier releases of CICS you would have to alter the cursor-sensing sections of the application to use only the EIBCPOSN field for determining the cursor positions.

All base colors are initialized in the maps, together with any required emphasis, and these are maintained by the application when it displays and removes pop-ups and pull-downs.

All maps are defined with literal values in mixed case.

In addition to normal entry field variables, fields which require color and emphasis attribute settings also require labels within the maps.

Any overlay pull-down or pop-up maps are sent to the screen with no ERASE option on the SEND call. As mentioned previously, an overlay map is defined with the unused area space filled. When the overlay map contains data entry fields, the underlying base map information shows through if the fields are not initialized. (This includes initializing any “stopper” fields.)

When you design a map to be overlaid on another, it is often impossible to determine which base map will be underneath, because the user path is not predefined by the application. Any fields in the base map which appear to the right of the overlay (that is, to the right border of the overlay) have their color attributes changed to the color of the overlay border. You can define the overlay map with a reset character following the right border, with a de-emphasized color attribute best suited to the bulk of the panels it overlays. This is a minor deviation from CUA but it ensures that all fields displayed to the right of the overlay will be in a de-emphasized color. Testing has shown that the user’s attention is focused on the overlay window, and the disruption to the base map is minimal.

The application obtains the cursor position in a map by symbolic cursor positioning. See the notes on “Moving the selection cursor” on page 15 for more information.

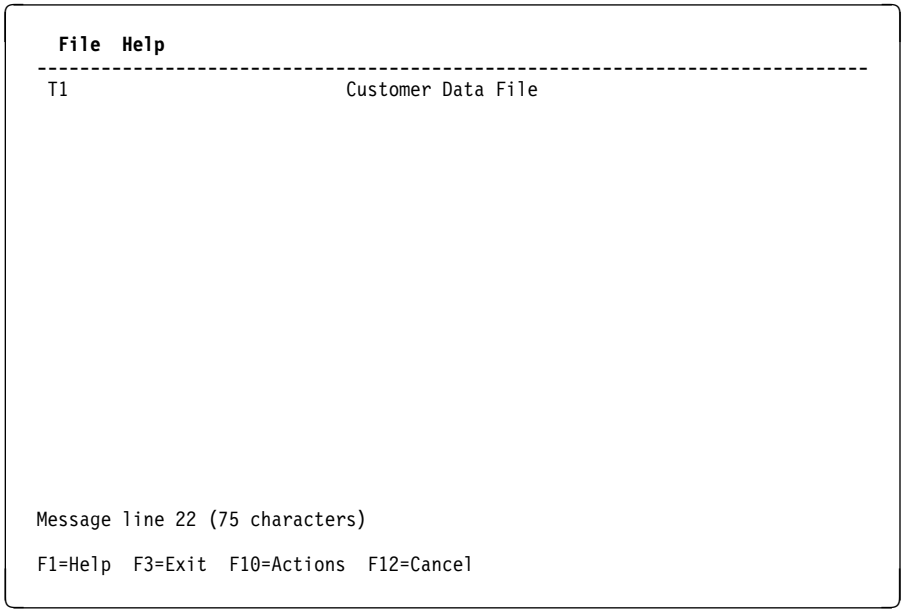
Some maps are redefined because the keywords GROUP and OCCURS are mutually exclusive in BMS. You must exercise care whenever modifying any redefined maps because the lengths must always match. A common cause of PROG 402 errors at the terminal is a map altered out of line with its redefined layout. This occurs mainly in the list panel area.

Warning and action messages are accompanied by an audible warning by using the SEND CONTROL ALARM call.

The following diagrams show the base maps and field names prior to any modification or attribute setting done by the programs.

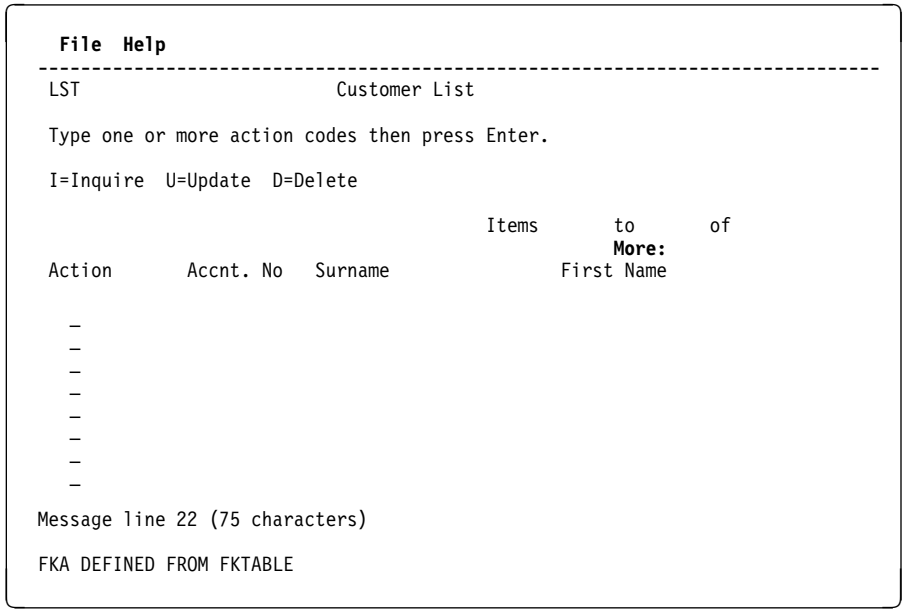
© Copyright IBM Corp. 1989,1999

Map T1: map set DFH0T1 (primary panel to sample application)



Field name	Map position	Notes
T1FFLD	01,03	File action
T1HFLD	01,09	Help action
T1ID	03,02	Panel ID
T1TITLE	03,30	Panel title, initial cursor position
T1MSG	22,01	Message line

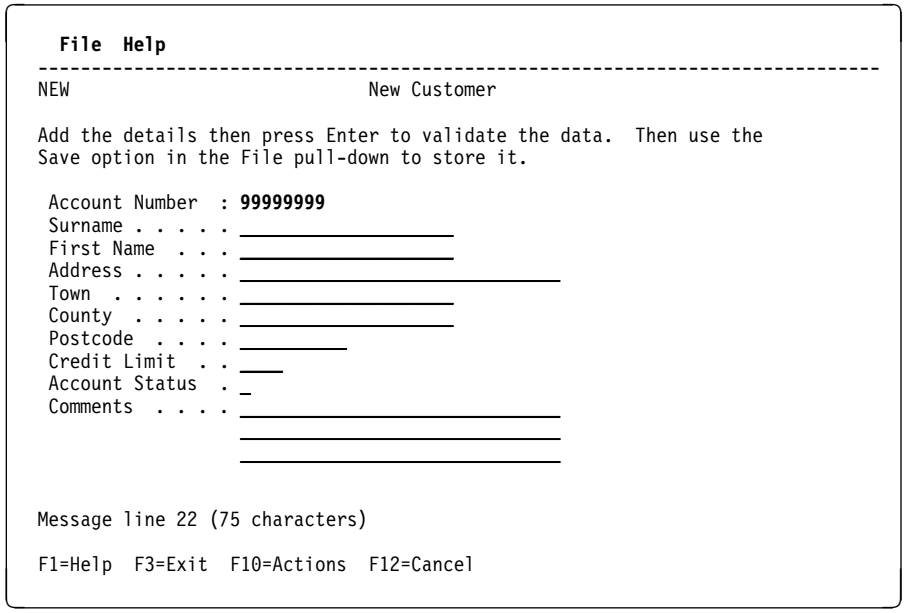
Map LST: map set DFH0LST (list processing - base panel)



Field name	Map position	Notes
LSTFFLD	01,03	File action
LSTHFLD	01,09	Help action
LSTID	03,02	Panel ID
FROMITM	09,50	From item
TOITEM	09,59	To item
OFITEM	09,68	Of item
MORE	10,56	More
MOREM	10,62	Minus sign
MOREP	10,64	Plus sign
ACTION1	13,04	Action line 1, initial cursor position
ACCTNO1	13,16	Account number line 1
SNAME1	13,26	Surname line 1
FNAME1	13,47	First name line 1
ACTMSG1	13,68	Action message line 1
ACTION2	14,04	Action line 2
ACCTNO2	14,16	Account number line 2
SNAME2	14,26	Surname line 2
FNAME2	14,47	First name line 2
ACTMSG2	14,68	Action message line 2
ACTION3	15,04	Action line 3
ACCTNO3	15,16	Account number line 3

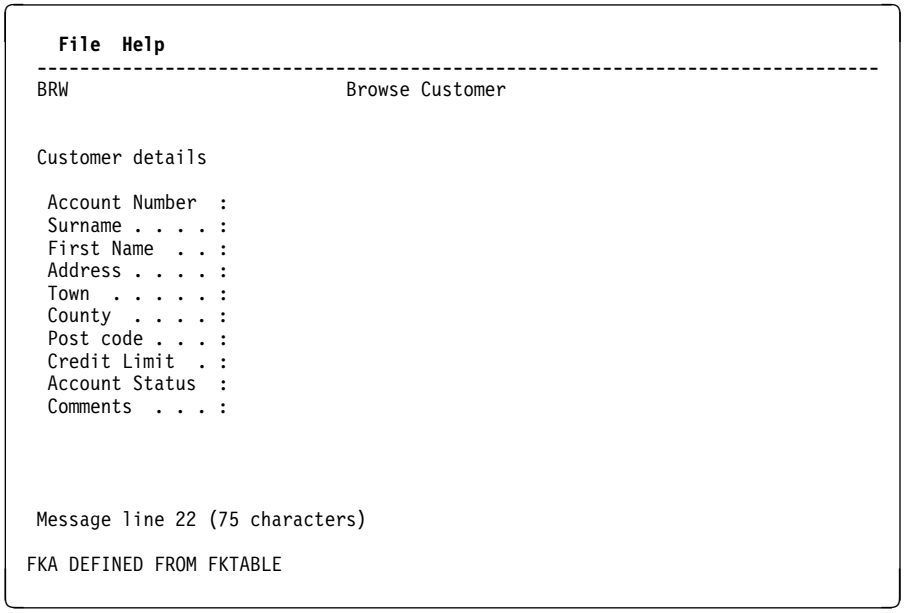
Field name	Map position	Notes
SNAME3	15,26	Surname line 3
FNAME3	15,47	First name line 3
ACTMSG3	15,68	Action message line 3
ACTION4	16,04	Action line 4
ACCTNO4	16,16	Account number line 4
SNAME4	16,26	Surname line 4
FNAME4	16,47	First name line 4
ACTMSG4	16,68	Action message line 4
ACTION5	17,04	Action line 5
ACCTNO5	17,16	Account number line 5
SNAME5	17,26	Surname line 5
FNAME5	17,47	First name line 5
ACTMSG5	17,68	Action message line 5
ACTION6	18,04	Action line 6
ACCTNO6	18,16	Account number line 6
SNAME6	18,26	Surname line 6
FNAME6	18,47	First name line 6
ACTMSG6	18,68	Action message line 6
ACTION7	19,04	Action line 7
ACCTNO7	19,16	Account number line 7
SNAME7	19,26	Surname line 7
FNAME7	19,47	First name line 7
ACTMSG7	19,68	Action message line 7
ACTION8	20,04	Action line 8
ACCTNO8	20,16	Account number line 8
SNAME8	20,26	Surname line 8
FNAME8	20,47	First name line 8
ACTMSG8	20,68	Action message line 8
LSTMSG	22,01	Message line
LSTFKA	24,01	Function key line

Map NEW: map set DFH0NEW (new customer record - base panel)



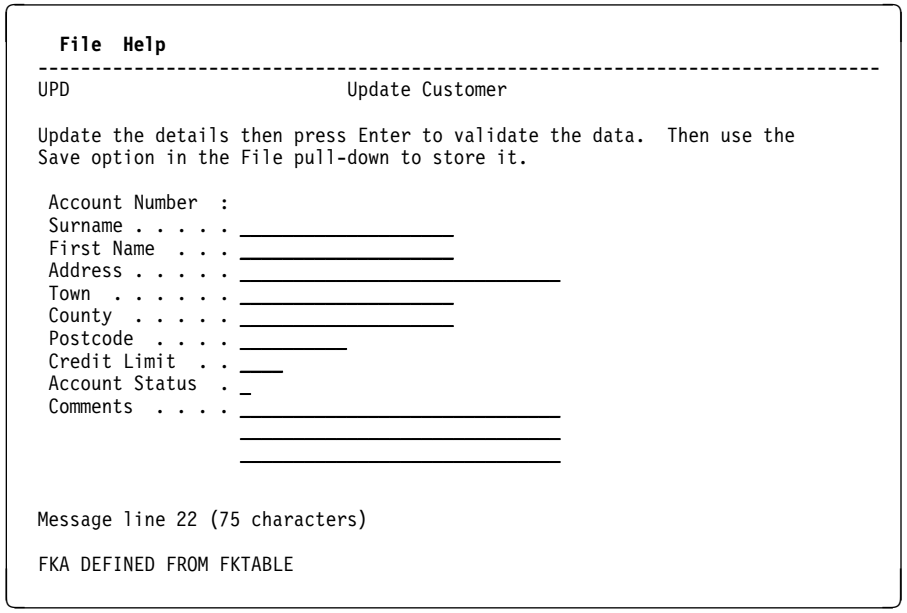
Field name	Map position	Notes
NEWFFLD	01,03	File action
NEWHFLD	01,09	Help action
NEWID	03,02	Panel ID
ACCNON1	08,21	Account number
SNAMEN1	09,21	Surname, initial cursor position
FNAMEN1	10,21	First name
ADDRN1	11,21	Address
TOWNN1	12,21	Town
COUNTN1	13,21	County
PCODEN1	14,21	Postcode
CRLIMN1	15,21	Credit limit
ACCSTN1	16,21	Account status
COMMN1	17,21	Comment line 1
COMMN2	18,21	Comment line 2
COMMN3	19,21	Comment line 3
NEWMSG	22,01	Message line

Map BRW: map set DFH0BRW (browse customer details - base panel)



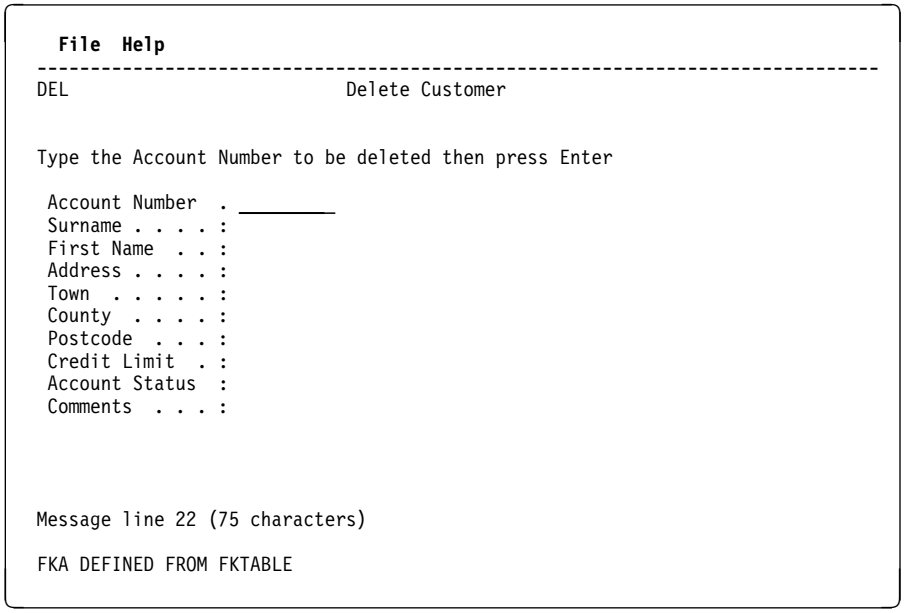
Field name	Map position	Notes
BRWFFLD	01,03	File action
BRWHFLD	01,09	Help action
BRWID	03,02	Panel ID
BRWTITL	03,30	Panel title, initial cursor position
ACCNOB1	08,21	Account number
SNAMEB1	09,21	Surname
FNAMEB1	10,21	First name
ADDRB1	11,21	Address
TOWNB1	12,21	Town
COUNTB1	13,21	County
PCODEB1	14,21	Post code
CRLIMB1	15,21	Credit limit
ACCSTB1	16,21	Account status
COMMB1	17,21	Comment line 1
COMMB2	18,21	Comment line 2
COMMB3	19,21	Comment line 3
BRWMSG	22,01	Message line
BRWFKA	24,01	Function Key line

Map UPD: map set DFH0UPD (update customer details - base panel)



Field name	Map position	Notes
UPDFFLD	01,03	File action
UPDHFLD	01,09	Help action
UPDID	03,03	Panel ID
ACCNOU1	08,21	Account number
SNAMEU1	09,21	Surname, initial cursor position
FNAMEU1	10,21	First name
ADDRU1	11,21	Address
TOWNU1	12,21	Town
COUNTU1	13,21	County
PCODEU1	14,21	Post code
CRLIMU1	15,21	Credit limit
ACCSTU1	16,21	Account status
COMMU1	17,21	Comment line 1
COMMU2	18,21	Comment line 2
COMMU3	19,21	Comment line 3
UPDMSG	22,01	Message line
UPDFKA	24,01	Function Key line

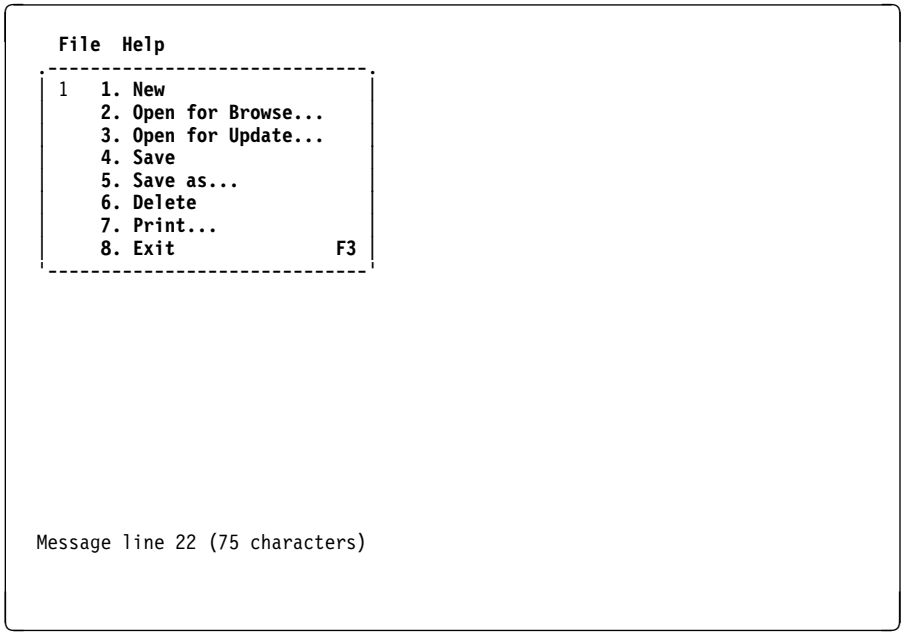
Map DEL: map set DFH0DEL (Delete a customer record - base panel)



Field name	Map position	Notes
DELFFLD	01,03	File action
DELHFLD	01,09	Help action
DELID	03,02	Panel ID
DELTITL	03,30	Panel title
COLOND1	08,19	Colon
ACCNOD1	08,21	Account number, initial cursor position
SNAMED1	09,21	Surname
FNAMED1	10,21	First name
ADDRD1	11,21	Address
TOWND1	12,21	Town
COUNTD1	13,21	County
PCODED1	14,21	Postcode
CRLIMD1	15,21	Credit limit
ACCSTD1	16,21	Account status
COMMD1	17,21	Comment line 1
COMMD2	18,21	Comment line 2
COMMD3	19,21	Comment line 3

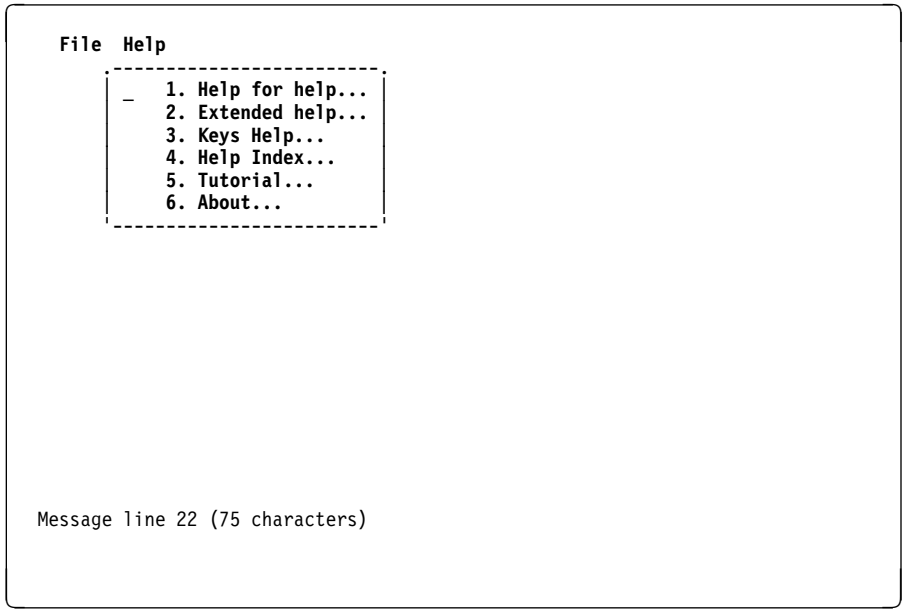
Field name	Map position	Notes
DELMSG	22,01	Message line
DELFKA	24,01	Function key line

Map FPD: map set DFH0FPD (file pull-down)



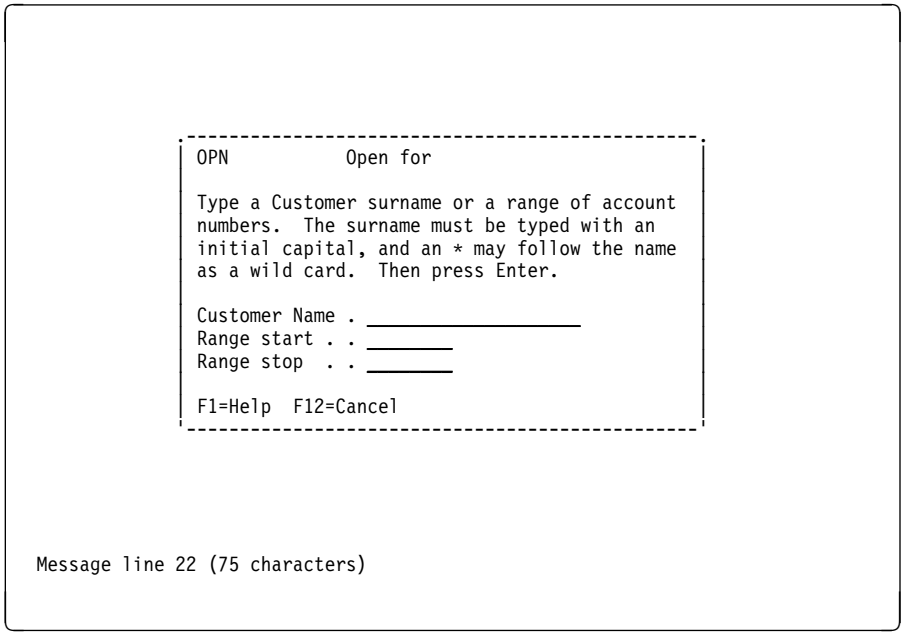
Field name	Map position	Notes
FPDFFLD	01,03	File action
FPDHFLD	01,09	Help action
FPDSEL	03,03	Selection field, initial cursor position
FPOPT1	03,07	Option 1 text
FPACT1	03,10	Action 1 text
FPOPT2	04,07	Option 2 text
FPACT2	04,10	Action 2 text
FPOPT3	05,07	Option 3 text
FPACT3	05,10	Action 3 text
FPOPT4	06,07	Option 4 text
FPACT4	06,10	Action 4 text
FPOPT5	07,07	Option 5 text
FPACT5	07,10	Action 5 text
FPOPT6	08,07	Option 6 text
FPACT6	08,10	Action 6 text
FPOPT7	09,07	Option 7 text
FPACT7	09,10	Action 7 text
FPOPT8	10,07	Option 8 text
FPACT8	10,10	Action 8 text
FPDMSG	22,01	Message line

Map HPD: map set DFH0HPD (help pull-down)



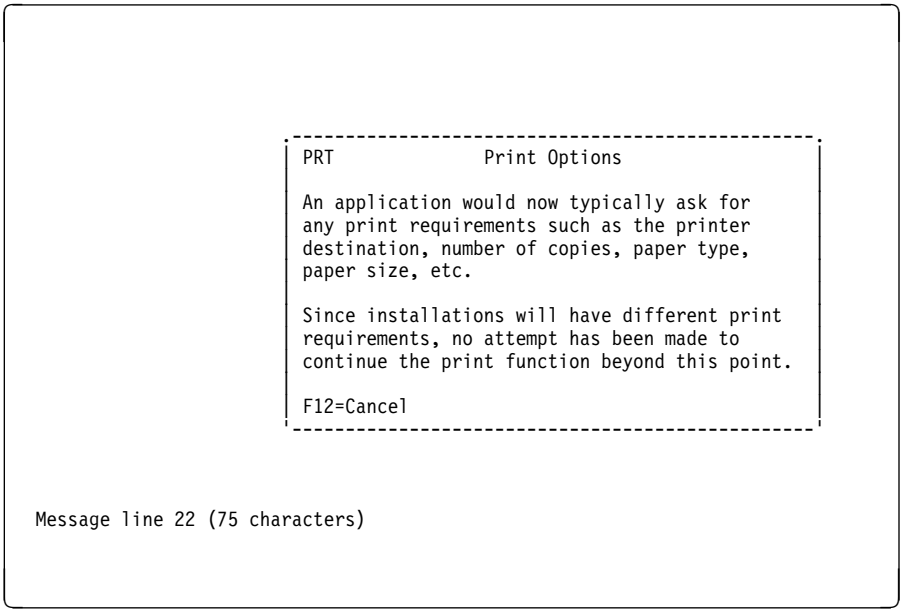
Field name	Map position	Notes
HPDFFLD	01,03	File action
HPDHFLD	01,09	Help action
HPDSEL	03,09	Selection field, initial cursor position
HPOPT1	03,13	Option 1 text
HPACT1	03,16	Action 1 text
HPOPT2	04,13	Option 2 text
HPACT2	04,16	Action 2 text
HPOPT3	05,13	Option 3 text
HPACT3	05,16	Action 3 text
HPOPT4	06,13	Option 4 text
HPACT4	06,16	Action 4 text
HPOPT5	07,13	Option 5 text
HPACT5	07,16	Action 5 text
HPOPT6	08,13	Option 6 text
HPACT6	08,16	Action 6 text
HPDMSG	22,01	Message line

Map OPN: map set DFH0OPN (file-open pop-up)



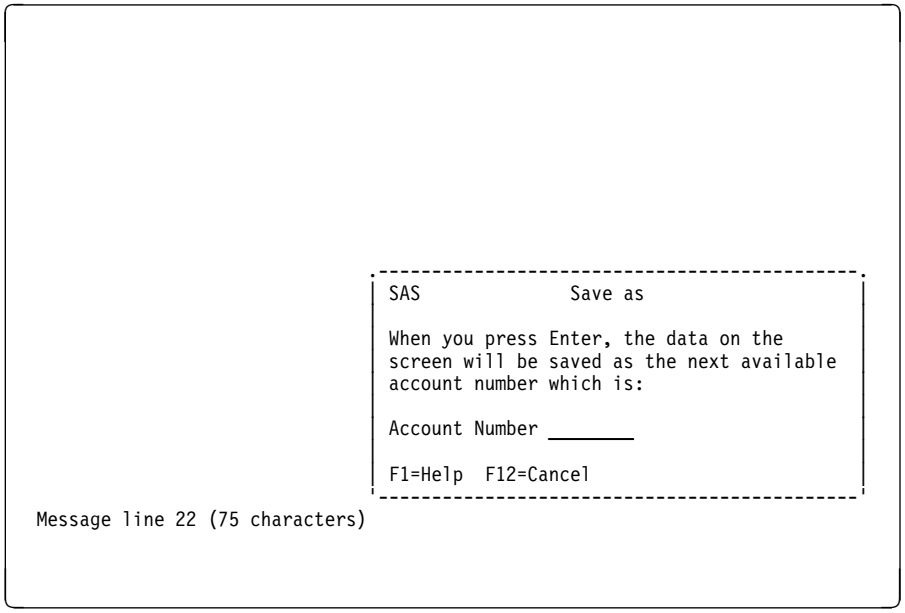
Field name	Map position	Notes
OPNMODE	06,41	Pop-up title
SNAMEO1	13,33	Surname, initial cursor position
RSTART	14,33	Range start
RSTOP	15,33	Range stop
OPNMSG	22,01	Message line

Map PRT: map set DFH0PRT (print pop-up)



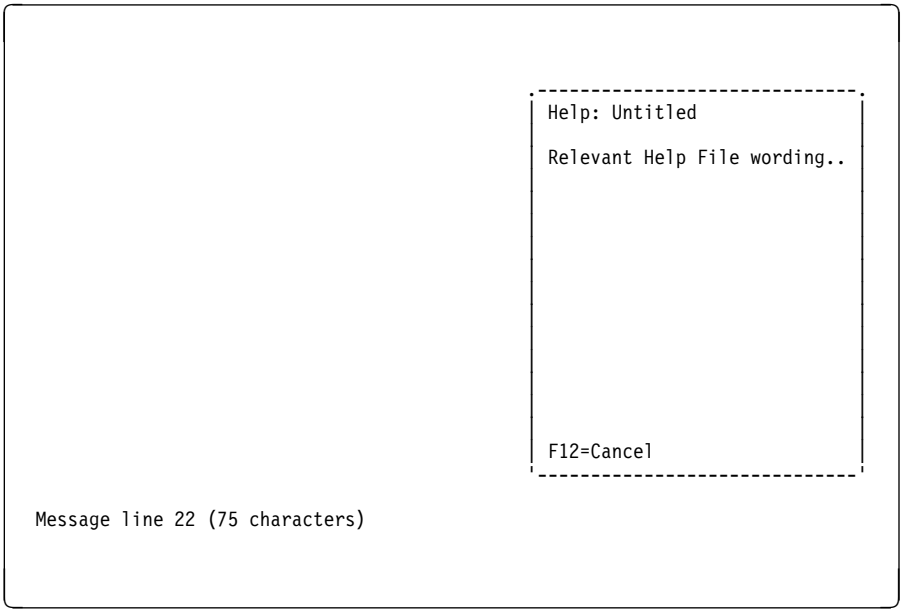
Field name	Map position	Notes
PRTTITL	06,44	Pop-up title, initial cursor position
PRTMSG	22,01	Message line

Map SAS: map set DFH0SAS (save changed customer record pop-up)



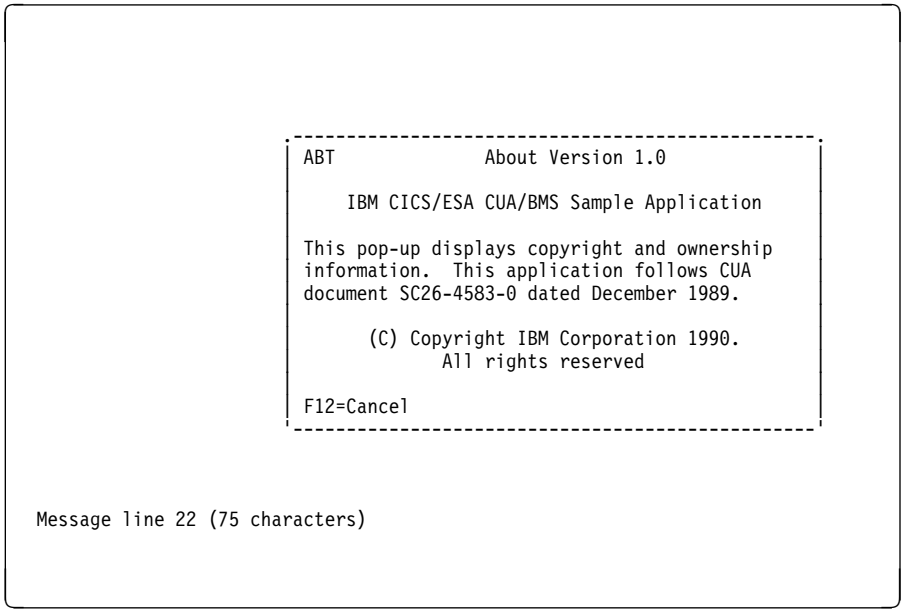
Field name	Map position	Notes
ACCNOS1	18,50	Account number, initial cursor position
SASMSG	22,01	Message line

Map HPOP: map set DFH0HP (contextual help pop-up)



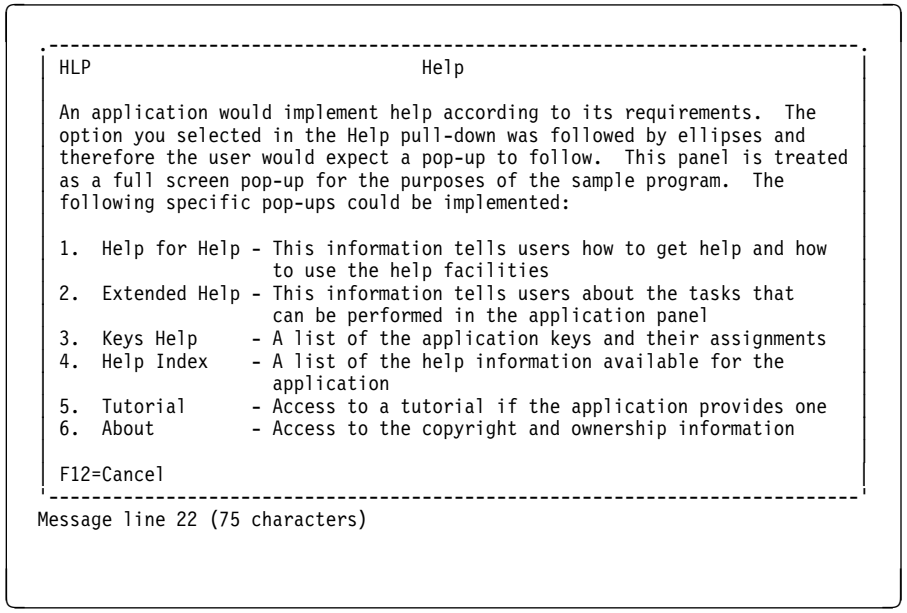
Field name	Map position	Notes
HTITLE	04,49	Pop-up title
HLINE1	06,49	Help text line 1, initial cursor position
HLINE2	07,49	Help text line 2
HLINE3	08,49	Help text line 3
HLINE4	09,49	Help text line 4
HLINE5	10,49	Help text line 5
HLINE6	11,49	Help text line 6
HLINE7	12,49	Help text line 7
HLINE8	13,49	Help text line 8
HLINE9	14,49	Help text line 9
HLINE10	15,49	Help text line 10
HLINE11	16,49	Help text line 11
HLINE12	17,49	Help text line 12
HPMSG	22,01	Message line

Map ABT: map set DFH0ABT (about the sample application pop-up)



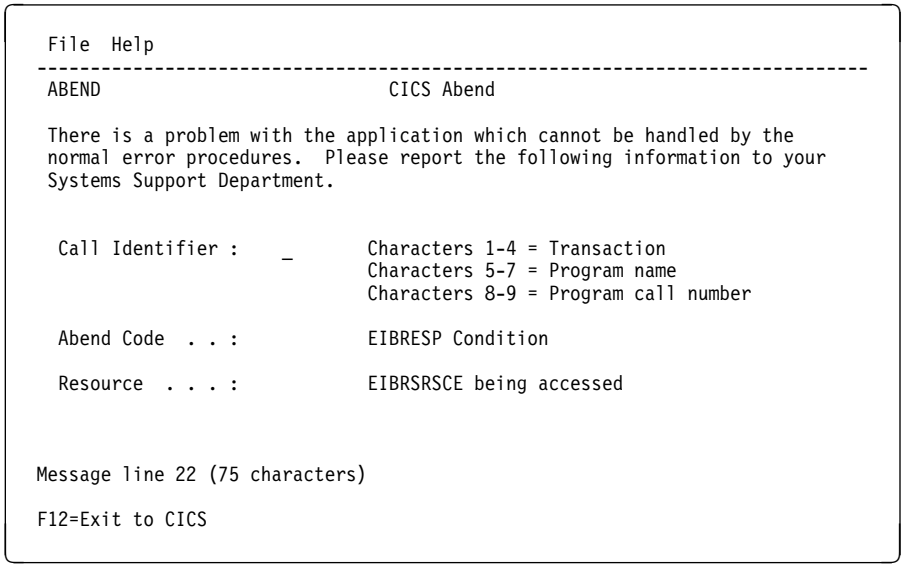
Field name	Map position	Notes
ABTTITL	06,43	Pop-up title, initial cursor position
ABTMSG	22,01	Message line

Map HLP: map set DFH0HLP (the help stub full screen pop-up)



Field name	Map position	Notes
HLPTITL	02,37	Panel title, initial cursor position
HLPMSG	22,03	Message line

Map AB: map set DFH0AB (abend handling)



Field name	Map position	Notes
ABID	03,02	Panel ID
ABTITLE	03,33	Panel title
ABCALL	11,21	Call identifier, initial cursor position
ABCODE	15,21	Abend code
ABRSRCE	17,21	Resource
ABMSG	22,01	Message line
ABFKA	24,01	Function key line

Part 2. FILEA sample applications

This part of the book describes the FILEA sample applications. There are four sets of command-level application programs which operate on FILEA, one in each of the programming languages Assembler, C/370™, VS COBOL II, and PL/I. Each set comprises six programs:

- Operator instruction
- Inquiry/update
- Browse
- Order entry
- Order entry queue print
- Low balance report.

These samples are provided as an Installation Verification Procedure (IVP). You should run them as soon as you bring up a new system. They serve as a useful test of whether you have successfully initialized a basic CICS system.

The programs were written prior to the publication of Common User Access (CUA) guidelines, but can be used as a basis for developing your own application programs.

This part of the book contains the following chapters:

- Chapter 12, “Installing and running the FILEA sample applications” on page 121
- Chapter 13, “FILEA sample application program descriptions” on page 125
- Chapter 14, “FILEA sample application file description” on page 131
- Chapter 15, “FILEA sample application BMS maps” on page 133.

© Copyright IBM Corp. 1989,1999

Chapter 12. Installing and running the FILEA sample applications

The resource definitions supporting the FILEA sample applications are provided in the CICS system definition file (CSD).

The applications are defined in the CSD groups generated by the INITIALIZE command of the CSD utility program, DFHCSDUP. For a list of the application groups in the CSD, and their contents, see the CICS Transaction Server for VSE/ESA Resource Definition Guide. Note that the groups of sample resource definitions are not included in the IBM-defined group list, DFHLIST. This means that, if you start CICS with GRPLIST=DFHLIST, you must install the sample groups using the CEDA INSTALL command before you can run any of the sample transactions.

The CICS resource table definitions you need to run these sample applications are provided in the following pregenerated sample control tables:

- **DFHDCT2$**, the sample destination control table, contains definitions for LOGA and L86O, two transient data queues.
- **DFHTCT5$**, the sample terminal control table, contains only the definitions for the sequential devices (CARDIN and PRINTER) needed to run the batch IVP, DFHIVPBT. Use CEDA to define the L86O printer, needed by the FILEA order entry print program, as a VTAM terminal.

You do not need to sign on (using CESN) to CICS in order to use the FILEA sample application programs. However, you should sign on, using CESN, with authority to use:

- The CEDF transaction if you want to run the sample programs under EDF, to test the CICS execution diagnostic facility
- The CECI transaction, if you want to access the application resources using the command level interpreter.

© Copyright IBM Corp. 1989,1999

Installing the sample groups

The transaction and program definitions for the sample applications are provided in the assembler language, C, COBOL, and PL/I group definitions in the CSD. Install these groups using the CEDA INSTALL command before trying to run any applications. The CEDA commands to install the sample groups are shown in Table 4.

Table 4. CEDA commands to install the sample groups

CEDA command	Description of resource definition
INSTALL GROUP(DFH$AFLA)	The assembler versions of the FILEA sample application programs
INSTALL GROUP(DFH$DFLA)	The C versions of the FILEA sample application programs
INSTALL GROUP(DFH$CFLA)	The COBOL versions of the FILEA sample application programs
INSTALL GROUP(DFH$PFLA)	The PL/I versions of the FILEA sample application programs
INSTALL GROUP(DFH$FILA)	The FILEA file resource definition (see note below)

Note: The FILEA file resource definition is in a separate group so that you can install the four language versions of the FILEA sample applications in the same run. If the file definition was in each of the FILEA sample application groups, only the first INSTALL GROUP would succeed, and the second and subsequent install commands for a group containing FILEA would then fail because of the installed status of FILEA.

As an alternative to installing groups using CEDA, you may prefer to modify the list of groups used in your GRPLIST startup parameter, so that they are automatically installed at CICS initialization. For details of the CEDA commands for creating and copying lists of groups, see the CICS Transaction Server for VSE/ESA Resource Definition Guide manual.

Language considerations

The only sample applications provided as pregenerated load modules ready for use are those written in assembler language. The assembler language sample maps are also provided ready to execute. These are supplied in PRD1.BASE, and all module names commence with DFH$.

However, the C, COBOL, and PL/I samples are provided in source form only in PRD1.BASE. Before you can run the C, COBOL, or PL/I programs, you must first assemble the maps and compile the programs for execution. The maps can be assembled and link-edited using the CICS-supplied procedure, DFHMAPS. Note that the map source member names are not the same as the map names defined in the programs. This allows you to store the symbolic description maps in the same library as the source member.

If you use DFHMAPS (sample JCL supplied) to prepare the BMS map sets needed for the sample programs, you must update the JCL with the name of the source file, map and library. Include the library in the libdef statement of your CICS JCL.

Some language considerations

You need LE/370 to compile the C, COBOL, and PL/I versions of the FILEA sample.

Running the sample applications

Once CICS is running, type the operator instruction needed for your language on to a clear screen and press the enter key. The operator instruction transaction identifier invokes the “Operator instruction” sample program, which is a short program that produces a menu containing the transaction identifiers for two of the other sample programs, namely “Inquiry/Update” and “Browse”.

If you clear the screen, remember to reenter the transaction identifier, because no data is accepted from an unformatted screen.

You do not need to sign on (using CESN) to CICS in order to use the sample application programs. However, you should sign on, using CESN, with authority to use:

- The CEDF transaction if you want to run the sample programs under EDF, to test the CICS execution diagnostic facility
- The CECI transaction, if you want to access the application resources using the command level interpreter.

From within the operator menu, you can invoke any of the transactions detailed in Table 5, by entering the four-character transaction identifier and the six-digit account number in the fields highlighted in the bottom line of the display.

These transaction identifiers give you access to the inquiry, add, and update functions of the “Inquiry/Update” program, and access to the “Browse” program.

Table 5. Sample programs and their identifiers

Language	Transaction identifiers				
	Operator instruction	**File inquiry**	**File browse**	**File add**	**File update**
Assembler	AMNU and NUMBER	AINQ and NUMBER	ABRW and NUMBER	AADD and NUMBER	AUPD and NUMBER
COBOL	MENU and NUMBER	INQY and NUMBER	BRWS and NUMBER	ADDS and NUMBER	UPDT and NUMBER
PL/I	PMNU and NUMBER	PINQ and NUMBER	PBRW and NUMBER	PADD and NUMBER	PUPD and NUMBER
C/370	DMNU and NUMBER	DINQ and NUMBER	DBRW and NUMBER	DADD and NUMBER	DUPD and NUMBER

Order entry and order entry queue print samples

You can invoke the three remaining sample programs “Order entry”, “Order entry queue print”, and “Low balance report” separately by entering their transaction identifiers (as shown in Table 6) on to a clear screen.

Table 6. Sample programs and their identifiers

Language	Transaction identifiers		
	Order entry	**Order entry queue print**	**Low balance report**
Assembler	AORD	AORQ	AREP
COBOL	OREN	OREQ	REPT
PL/I	PORD	PORQ	PREP
C/370	DORD	DORQ	DREP

To run the Order entry queue print sample, you must have defined the L86O and LOGA transient data queues (TDQs). You can define a display terminal to receive the output from the L86O queue. First, find out the termtype for your display (see “Switching off uppercase translation at the terminal level” on page 65 for details of how to do this). You can then use the CEDA transaction to define the L86O as a terminal of the relevant termtype, with a suitable display address.

Chapter 13. FILEA sample application program descriptions

The sample programs described in this chapter are included, in the CICS distribution tape, in both source and processable form for assembler language, and in source form only for COBOL, PL/I, and C. Chapter 12, “Installing and running the FILEA sample applications” on page 121 describes how these sample programs, and associated resources, can be defined to CICS.

This chapter describes six CICS sample application programs, written in assembler language, COBOL, PL/I, and C, as follows:

- Operator instruction
- Inquiry/update
- Browse
- Order entry
- Order entry queue print
- Low balance report.

These programs show basic applications (such as inquire, browse, add, and update) that can serve as a framework for your installation’s first programs. Each program has a short description of what the program does, a listing of its source code, and a series of program notes. Numbered coding lines in the source listing correspond to the numbered program notes. In COBOL, the programs contain COPY statements coded according to the 1968 COBOL standard.

These sample programs are for use with the IBM 3270 Information Display System and were written prior to the publication of Common User Access guidelines.

Operator instruction sample program (3270)

The instruction program displays a map containing operator instructions. This map lists some of the sample application programs provided for the programming language being used, and the transaction identifiers that can be used to invoke them. To initiate the browse, add, update, or inquiry programs, the appropriate transaction identifier must be entered on the menu map.

The program names, map names, and transaction identifiers for this function are:

Language	Program name	Map names	Trans. Id
Assembler	DFH$AMNU	DFH$AGA	AMNU
COBOL	DFH0CMNU	DFH0CGA	MENU
PL/I	DFH$PMNU	DFH$PGA	PMNU
C/370	DFH$DMNU	DFH$DGA	DMNU

© Copyright IBM Corp. 1989,1999

Inquiry/update sample program (3270)

The inquiry/update sample program lets you make an inquiry about, add to, or update records in a file. You can select one of these actions by entering the appropriate transaction identifier (see Table 5 on page 123) in the menu that is displayed when you start operations by entering the operator instruction.

To make an inquiry, enter the transaction identifier for the inquiry transaction, and an account number into the menu. The program maps in the account number and reads the record from FILEA. The required fields from the file area, and a title "FILE INQUIRY" are moved to the map DSECT, containing the record fields, and are displayed at your screen.

The program names, map names, and transaction identifiers for this function are:

Language	Program name	Map names	Trans. Id
Assembler	DFH$AALL	DFHAGA,DFHAGB	AINQ,AADD,AUPD
COBOL	DFH0CALL	DFH0CGA,DFH0CGB	INQY,ADDS,UPDT
PL/I	DFH$PALL	DFHPGA,DFHPGB	PINQ,PADD,PUPD
C/370	DFH$DALL	DFHDGA,DFHDGB	DINQ,DADD,DUPD

To add a record, enter the add transaction identifier (see Table 5 on page 123) and the account number into the menu. The account number and a title "FILE ADD" are moved to the map area of the map DSECT, containing empty data fields. This is displayed at your screen. The data fields entered are mapped into the map DSECT and moved to the file record area which is then written to FILEA. The addition is recorded on an update log (LOGA), which is a transient data queue. The operator instruction screen is displayed with the message "RECORD ADDED."

To update a record, enter the relevant transaction (see Table 5 on page 123) and the account number into the menu, as before. The program reads and displays the requested FILEA record. Modified data fields are mapped in to the map area DSECT and edited. The sample program only suggests the type of editing you might want to do. You should insert editing steps needed to ensure valid changes to the file. Those fields which have been changed are moved to the data record and the record is rewritten to FILEA. The update is recorded on LOGA. The message "RECORD UPDATED" is moved to the DSECT for the relevant map, the operator instruction menu map, which is then displayed at your screen.

This program is an example of pseudoconversational programming, in which control is returned to CICS together with a transaction identifier whenever a response is requested from the operator. Associated with each return of control to CICS is a storage area containing details associated with the previous invocation of this transaction.

Browse sample program (3270)

The browse program sequentially retrieves a page or set of records for display, starting at a point in a file specified by the terminal operator.

To start a browse, type the transaction identifier (see Table 5 on page 123) and an account number into the menu and press the Enter key. If you omit the account number, browsing begins at the start of the file. Pressing the PF1 key or typing F

causes retrieval of the next page or paging forward. If you want to reexamine the previous records displayed, press PF2 or type B. This lets you page backward.

The browse program uses READNEXT to page forward to the end of the file and READPREV to page backward to the start of the file.

The program names, map names, and transaction identifiers for this function are:

Language	Program name	Map names	Trans. Id
Assembler	DFH$ABRW	DFHAGA,DFHAGC	ABRW
COBOL	DFH0CBRW	DFH0CGA,DFH0CGC	BRWS
PL/I	DFH$PBRW	DFHPGA,DFHPGC	PBRW
C/370	DFH$DBRW	DFHDGA,DFHDGC	DBRW

Order entry sample program (3270)

The order entry sample application program provides a data entry facility for customer orders for parts from a warehouse. Orders are recorded on a transient data queue that is defined so as to start the order entry queue print transaction automatically when a fixed number of orders have been accumulated. The queue print transaction sends the orders to a printer terminal at the warehouse.

To begin order entry, type the transaction identifier (see Table 6 on page 124) on to a blank screen and press ENTER.

The program names, maps, and transaction identifiers for this function are:

Language	Program name	Map names	Trans. Id
Assembler	DFH$AREN	DFH$AGK	AORD
COBOL	DFH0CREN	DFH0CGK	OREN
PL/I	DFH$PREN	DFH$PGK	PORD
C/370	DFH$DREN	DFH$DGK	DORD

The order entry program displays the map on the screen, as shown above, requesting the operator to enter order details, that is, customer number, part number, and the quantity of that part required. The customer number must be valid, that is, it must exist on FILEA. The order details are mapped in and checked; an invalid order is redisplayed for correction. When valid, an order is written to the transient data queue L86O and the order entry screen is redisplayed ready for the next order to be entered. If CLEAR is pressed the order entry program terminates.

L86O, the name of the transient data queue, is also the name of the terminal where the order entry queue print transaction is to be triggered when the number of items on the queue reaches 30. A definition of the transient data queue is included in the sample destination control table listed in the CICS Transaction Server for VSE/ESA System Definition Guide. Note that if you are using COBOL, PL/I, or C, the TRANSID specified in the DCT entry for L86O must be changed from AORQ to the transaction identifier so that your language program can be triggered.

The trigger level may be changed using the CEMT command, as follows:

```
CEMT SET QUEUE(L86O) TRIGGER(n)
```

where n is the destination trigger level (any integer from 0 through 32767).

Order entry queue print sample program (3270)

The order entry queue print sample program sends customer orders to a printer terminal at the warehouse. The order entry sample program, described earlier, records customer orders on a transient data queue that is read by this program.

The queue print transaction can be invoked in one of three ways:

1. You can type the transaction identifier (see Table 6 on page 124) on to a clear screen. The program finds that the terminal identifier is not L86O and issues a START command to begin printing in one hour. The message “PROCESSING COMPLETED” is displayed and your terminal is available for other work.
2. One hour after you enter this transaction identifier the queue print transaction is automatically invoked by CICS interval control. In this case the terminal identifier, specified by the START, is L86O so the program prints the orders at the warehouse.
3. The queue print transaction is “triggered” when the number of items (customer orders) on the transient data queue reaches 30. The trigger level is specified in the destination control table (DCT) entry for L86O. In this case the terminal identifier is the same as the queue name (L86O) and the program prints the orders. Note that if you are using PL/I, COBOL, or C, the TRANSID specified in the DCT entry for L86O must be changed from AORQ to the transaction identifier for the language of your program to be triggered. The trigger level may be changed using the command:

   ```
   CEMT SET QUEUE(L86O) TRIGGER(n)
   ```

When invoked with a terminal identifier of L86O, the program reads each order, checks the customer’s credit, and either prints the order at the warehouse or writes the rejected order to LOGA, the same transient data queue as used by the inquiry/update sample program. When all the orders have been processed, or if there were no orders to process, the message “ORDER QUEUE IS EMPTY” is printed at the warehouse.

The program names, map names, and transaction identifiers for this function are:

Language	Program name	Map names	Trans. Id
Assembler	DFH$ACOM	DFH$AGL	AORQ
COBOL	DFH0CCOM	DFH0CGL	OREQ
PL/I	DFH$PCOM	DFH$PGL	PORQ
C/370	DFH$DCOM	DFH$DGL	DORQ

Low balance report sample program (3270)

The low balance report sample program produces a report that lists all entries in the data set FILEA for which the amount is less than or equal to $50.00.

The program illustrates page-building techniques and the use of the terminal paging facilities of BMS.

The transaction is invoked by entering the transaction identifier (see Table 6 on page 124) on to a clear screen. The program does a sequential scan through the file, selecting each entry that obeys the search criteria.

The program names, map names, and transaction identifiers for this function are:

Language	Program name	Map names	Trans. Id
Assembler	DFH$AREP	DFH$AGD	AREP
COBOL	DFH0CREP	DFH0CGD	REPT
PL/I	DFH$PREP	DFH$PGD	PREP
C/370	DFH$DREP	DFH$DGD	DREP

The pages are built from four maps that comprise the map set as described in the table above, using the paging option so that the data is not displayed immediately but, instead, is stored for later retrieval. The HEADING map is inserted at the head of each page. This detail map is written repeatedly until the OVERFLOW condition occurs. The FOOTING map is then written at the foot of the page and the HEADING map written at the top of the next page. The command to write the detail map that caused overflow is then repeated. When all the data has been written the FINAL map is written at the bottom of the last page and the transaction terminated.

The terminal operator then enters paging commands to display the data, clearing the screen before entering each paging command.

The program illustrates page-building techniques and use of the terminal paging facilities of BMS. The following paging commands are defined in the sample system initialization tables:

```
PGRET=P/
PGPURGE=T/
PGCOPY=C/
PGCHAIN=X/
```

Chapter 14. FILEA sample application file description

The sample programs operate using a VSAM file, known as FILEA, consisting of records containing details of individual customer accounts. This file must first be defined using the VSAM utility, IDCAMS. DFH$FAIN in PRD1.BASE contains the JCL needed to create and load the included data for this file. After the load, you will find that the file consists of records containing details of individual bank accounts, and the record key is the six-digit account number. The accounts on the file include account numbers 100000, 111111, 200000, 222222, 300000, 333333, 400000, 444444, 500000, 555555, 600000, 666666, 700000, 777777, 800000, 888888, 900000, and 999999.

The JCL statements assume that the VSAM user catalog already exists.

The FILEA VSAM file is also used in the Intercommunication sample applications.

© Copyright IBM Corp. 1989,1999

Chapter 15. FILEA sample application BMS maps

The sample BMS maps include examples of how the COLOR, EXTATT, and HILIGHT attributes are specified in the map definition macros. However, because of production limitations, the associated screen layouts do not show you all the effects of these attributes.

You can add attributes without changing the application program by specifying EXTATT=MAPONLY in the DFHMSD map set definition macro. If you include an attribute that specifies a facility not available at the terminal, it is ignored.

The names of the map source members in PRD1.BASE for the FILEA sample programs are not the same as the map names defined in the programs. This allows you to store the symbolic description maps in the same library as the source member.

The sixth letter is always “M” in the map source member name, and ”G” in the map names specified in the application programs. For example, the maps used in the update sample programs are named as follows:

Language	Program name	Map names in program	Source map names
Assembler	DFH$AALL	DFHAGA, DFHAGB	DFHAMA, DFHAMB
COBOL	DFH0CALL	DFH0CGA, DFH0CGB	DFH0CMA, DFH0CMB
PL/I	DFH$PALL	DFHPGA, DFHPGB	DFHPMA, DFHPMB
C/370	DFH$DALL	DFHDGA, DFHDGB	DFHDMA, DFHDMB

© Copyright IBM Corp. 1989,1999

Part 3. Intercommunication sample applications

This part of the book describes the CICS-supplied application programs that illustrate the use of distributed transaction processing and asynchronous processing on APPC and LUTYPE6.1 links.

The four applications in this group of samples, all of which are written in assembler language, are:

1. Transferring a temporary storage queue from a local CICS system to a remote CICS system, using distributed transaction processing and APPC protocols.
2. Browsing a remote file, using distributed transaction processing and APPC protocols.
3. Retrieving a record from a remote temporary storage queue, using asynchronous processing. This sample can be used with APPC and LUTYPE6.1 links.
4. Enabling a CICS-to-remote LUTYPE6.1 system conversation. LUTYPE6.1 links must be used for this sample.

This part of the book contains the following chapter:

- Chapter 16, “The intercommunication sample applications” on page 137.

© Copyright IBM Corp. 1989,1999

Chapter 16. The intercommunication sample applications

The intercommunication sample programs and their associated BMS mapsets are provided in both source and object form on the CICS distribution volume.

Intercommunication sample programs: The source modules have the same names as their equivalent object modules, and can be found in PRD1.BASE. For all the sample programs, the transaction name is the last four characters of the module name.

Intercommunication sample mapsets: The source modules, the symbolic description maps, and the physical maps can be found in PRD1.BASE. All the BMS mapset source modules have names of the form DFH$IMx. The symbolic description map and the physical map generated from DFH$IMx are both named DFH$IGx.

Resource definition: All the transaction, program, and mapset definitions for the Intercommunication sample programs are provided in the CICS-supplied group DFH$ICOM.

The names of the source modules for the sample programs and mapsets are shown in Table 7.

Table 7. Summary of sample application programs

Sample	Description	Source modules	Module description
1	Transfer of a temporary storage queue to a remote CICS system.	DFH$IQXL DFH$IQXR DFH$IMX	Local transaction Remote transaction BMS mapset DFH$IGX
2	Browsing a remote file.	DFH$IFBL DFH$IFBR DFH$IMB	Local transaction Remote transaction BMS mapset DFH$IGB
3	Retrieval of a record from a remote temporary storage queue.	DFH$IQRL DFH$IQRR DFH$IQRD DFH$IM1 DFH$IM2	Local request transaction Remote retrieve transaction Local display transaction BMS mapset DFH$IG1 BMS mapset DFH$IG2
4	CICS-to-CICS conversation	DFH$ICIC DFH$IMC	Local/remote transaction BMS mapset DFH$IGC

© Copyright IBM Corp. 1989,1999

Intercommunication sample 1 – temporary storage queue transfer

This sample illustrates the use of distributed transaction processing to transmit a temporary storage queue to a remote system. It consists of a front-end transaction (DFH$IQXL), a back-end transaction (DFH$IQXR), and a BMS mapset (DFH$IMX) that is used by the front-end transaction.

The front-end transaction is invoked by the transaction code IQXL, and displays the following menu at the user's terminal:

```
                    CICS-CICS QUEUE TRANSFER
                      SAMPLE PROGRAM MAP
                    *************************


                  LOCAL TS Q NAME ...
                  REMOTE TS Q NAME ..  REMOTEQ
                  REMOTE SYSTEM ID ..


                  TYPE IN VALUES, THEN PRESS ENTER
                  OR HIT "PF3" TO TERMINATE.
```

The displayed menu has three input fields:

LOCAL TS Q NAME
Specifies the name of the local temporary storage queue that is to be transferred to the remote system.

If this field is left blank, the front-end transaction builds for itself a small (5 records) temporary storage queue to transfer to the remote system.

REMOTE TS Q NAME
Specifies the name that the transferred queue is to be given on the remote system.

The menu supplies the default name REMOTEQ.

REMOTE SYSTEM ID
Specifies the name of the remote system.

This name must be the connection name of an APPC link.

The front-end transaction initiates the back-end transaction and transmits the temporary storage records for writing on the remote queue.

The user is informed of data input errors, and also of the progress of the queue transfer operation. The local temporary storage queue is deleted after successful completion.

Figure 8 on page 139 shows the overall flow of the queue-transfer sample.

```
         Local transaction                Remote transaction
           (front-end)                      (back-end)

1. . . . Get user requirements
2. . . . Create TS queue if needed
3. . . . ALLOCATE SYSID( )
         (MVC ATCHSESS,EIBRSRCE)
4. . . . CONNECT PROCESS
5. . . . SEND remote queue name  =====>   RECEIVE queue name
6. . . . SEND record (loop)      =====>   RECEIVE record (loop)
7. . . . SYNCPOINT               =====>
                                <=====    SYNCPOINT
8. . . . FREE SESSION                     RETURN
9. . . . DELETE TS queue
         RETURN
```

1. The user's input values are received and are validated.
2. If a local queue name is not supplied, a queue is constructed.
3. The front-end transaction allocates a conversation and acquires its name from the EIB.
4. The back-end transaction is initiated, using a SYNCLEVEL of 2 to allow CICS syncpointing.
5. The name of the remote queue is transmitted to the back-end transaction.
6. Using consecutive sends, the front-end transaction sends one queue record at a time to the back-end transaction, until the end of the queue is reached.

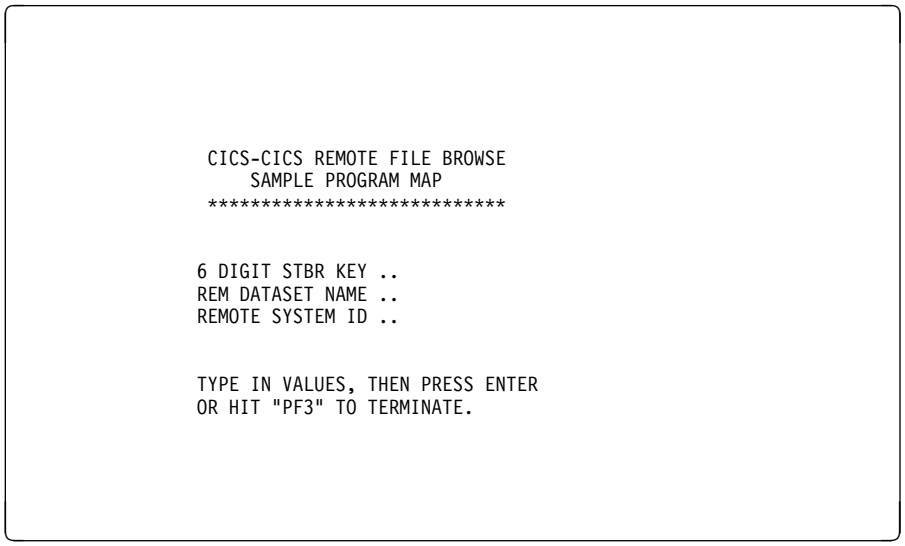
   The back-end transaction receives one record at a time and writes it to the temporary storage queue.
   The end of the transfer is indicated by the EIB settings.
7. When all the records have been sent, the front-end transaction issues a syncpoint.
   The back-end transaction, on checking the EIB, does the same.
8. The front-end transaction frees the session.
   The back-end transaction terminates (thus freeing the session) when the EIB shows FREE.
9. Finally, the front-end transaction deletes the local temporary storage queue and terminates.

Figure 8. Sample 1: Temporary storage queue transfer - overall design

Intercommunication sample 2 – remote file browse

This intercommunication sample illustrates the use of distributed transaction processing to browse a remote file. It consists of a front-end transaction (DFH$IFBL), a back-end transaction (DFH$IFBR), and a BMS mapset (DFH$IMB) for the front-end transaction.

The front-end transaction is invoked by the transaction code IFBL, and displays the following menu at the user's terminal:



The displayed menu has three input fields:

6 DIGIT STBR KEY
Specifies the key of the record at which the browse is to start.

REM DATASET NAME
Specifies the name of the data set that is to be browsed.

REMOTE SYSTEM ID
Specifies the name of the remote system. This name can be the connection name of an APPC link, an LUTYPE6.1 link, or an MRO link.

Initially, the file is browsed forwards, and four records from the remote file are displayed. Thereafter, the user can choose to browse forwards (PF8), browse backwards (PF7), or terminate the browse (PF3).

Figure 9 on page 141 shows the overall flow of the file browse sample.

```
          Local transaction                Remote transaction
            (front-end)                      (back-end)

 1. . . . Get user requirements
 2. . . . ALLOCATE SYSID( )
          (MVC ATCHSESS,EIBRSRCE)
 3. . . . CONVERSE                =====>   RECEIVE
 4. . . .                                   Process and buffer records
 5. . . .                         <=====    SEND
 6. . . .                                   RETURN
 7. . . . FREE conversation
 8. . . .  Process input and
           send to user's screen
 9. . . .  If more browsing go to 2
10. . . . RETURN
```

1. The user's input values are received and are validated.
2. The front-end transaction allocates a conversation and acquires its name from the EIB.
3. The data set name, the record key, and the browse direction are sent to the back-end transaction.

   Note that the CONVERSE command is equivalent to SEND, WAIT, RECEIVE. This is a “migration mode” command.
   No CONNECT PROCESS command is issued; instead, the remote transaction identifier is sent in the first four bytes of the message.
4. The back-end transaction retrieves four records from the data set and places them in a buffer.
5. The back-end transaction sends the buffered records to the front-end transaction, together with the key of the last record that was retrieved.
6. The back-end transaction terminates, thereby freeing the session.
7. The front-end transaction frees the session after receiving the data.
8. The front-end transaction unblocks the records and sends them (or possibly an error message) to the user.
9. If the user wants to browse more records (PF7 or PF8), the process is repeated from step 2.
10. Otherwise, the front-end transaction terminates.

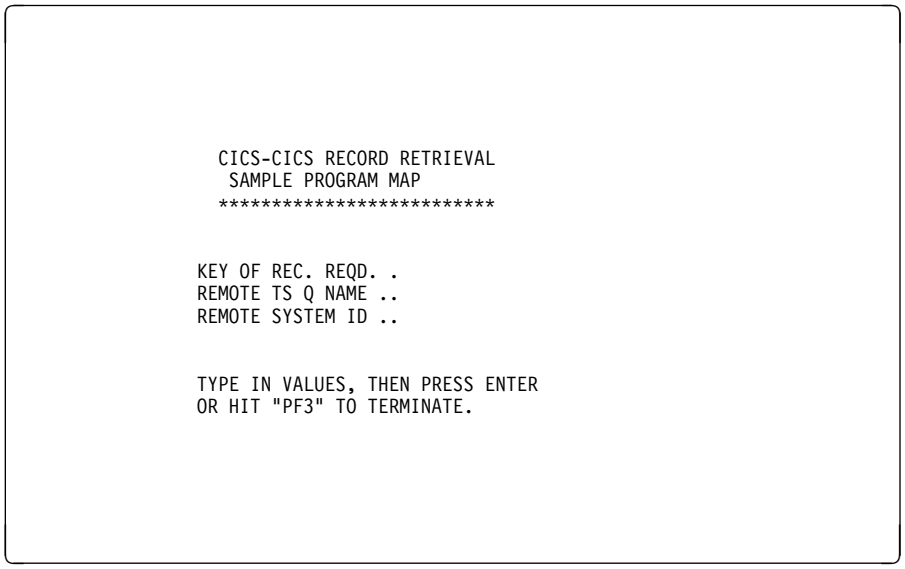
Figure 9. Sample 2: remote file browse - overall design

Intercommunication sample 3 – remote record retrieval

This intercommunication sample illustrates the use of asynchronous processing to retrieve a single record from a remote temporary storage queue. It consists of a local transaction (IQRL) to send the request to the remote system, a remote transaction (IQRR) to retrieve the record and return it to the local system, and a local transaction (IQRD) to receive the record and display it at the user terminal.

The remote temporary storage queue is assumed to consist of records that have unique user-defined keys in their first six bytes. If you want to run this sample, you will have to create a temporary storage queue of this form on the remote system.

The request transaction is invoked by the transaction code IQRL, and displays the following menu at the user's terminal:



KEY OF REC REQD
Specifies the user-defined key (that is, the first six bytes of data) of the remote temporary storage record.

REMOTE TS Q NAME
Specifies the name of the remote queue from which the record is to be retrieved.

REMOTE SYSTEM ID
Specifies the name of the remote system.

This name can be the connection name of an APPC, an LUTYPE6.1 link, or an MRO link.

The local request transaction uses a START command to start the remote retrieve transaction. The start request passes the name of the queue, the record number, the return transaction identifier (IQRD), and the return terminal identifier (obtained from the EIB).

It also passes the APPLID of the local CICS system. This enables the remote transaction to find the SYSID of the system that issued the initial start request. Because both the local and the remote transactions name a SYSID explicitly on their START commands, neither of the systems requires a remote transaction definition.

The remote transaction retrieves the required record, and passes it back to the local system, again by means of a START command. This START command names the local display transaction IQRD.

The local display transaction then displays the record at the user's terminal.

Intercommunication sample 4 – CICS to CICS conversation

The CICS-to-CICS synchronous sample application program allows a terminal operator to enter a command on the screen and have that command transmitted to a remote system for execution.

The front-end transaction is invoked by the transaction code ICIC, and displays the following menu at the user's terminal:

```
TYPE REMOTE SYSTEM ID AND COMMAND

REMOTE SYSTEM ID
COMMAND

THEN PRESS ENTER TO CONTINUE, OR
CLEAR TO TERMINATE
```

The program is able to converse with any application on a remote system that sends output data either one line at a time or in multiple line format. The CICS-supplied programs listed below have this capability; thus this example provides the CICS system programmer with a simple test transaction that proves it is easily possible to establish contact with a second, remote CICS system without the need for any application program coding. A successful test of this sample indicates, to the extent of the features actually being tested, that the system network has been correctly set up and that the intersystem components of CICS to allow distributed transaction processing are in order; failure indicates errors in setup rather than in user programming.

At the start of the program, the operator is prompted to enter the name of the remote system to be attached, and the actual command to be executed on the remote system, which is entered just as if it were a local command. The program can handle both single line output from the remote system and also output that exceeds the terminal page size.

The message received from the remote system is assumed to be in SCS form, that is, containing printable characters and new line symbols only. This is the default output format for LU6 type terminals as produced by CICS-supplied programs such as CSFE, CEMT, CEOT, or CEST.

Source listing of sample 4, combined front-end and back-end transaction (DFH$ICIC)

The numbers within this listing, 1), 2), and so on, refer to the program notes, which can be found at the end of the listing.

```
         TITLE 'DFH$ICIC - INTERCOMMUNICATION SAMPLE - CICS TO CICS OR  *
               IMS CONVERSATION'
DFHEISTG DSECT
*        STORAGE AREA FOR EIB SESSION AND STATUS FLAGS
XDFEIFLG DS    0CL7
XSYNC    DS    C                   IF SET, SYNCPOINT MUST
*                                  BE EXECUTED
XFREE    DS    C                   IF SET, TERMINAL / LU
*                                  MUST BE FREED
XRECV    DS    C                   IF SET, RECEIVE MUST
*                                  BE EXECUTED
XSEND    DS    C                   RESERVED
*
XATT     DS    C                   IF SET, ATTACH HEADER
*                                  DATA EXISTS AND MAY BE
*                                  ACCESSED USING EXTRACT
XEOC     DS    C                   IF SET, END-OF-CHAIN
*                                  WAS RECEIVED WITH DATA
XFMH     DS    C                   IF SET, DATA PASSED TO
*                                  APPL'N CONTAINS FMH(S)
         COPY  DFH$IGC               COPY MAP
         COPY  DFHAID
REMDATA  DS    256D
ATCHSESS DS    CL4
CONTROL  DS    0CL60
SBA      DS    CL3
CDATA    DS    CL57
MESSAGE  DS    CL32
INLEN    DS    H
OUTLEN   DS    H
RESP     DS    F
NEWLINE  EQU   X'15'
         EJECT
DFH$ICIC CSECT
MAPFAIL  XC    MAPAI(MAPAE-MAPAI),MAPAI    CLEAR MAP
 1)      EXEC CICS SEND MAP('MAPA') MAPSET('DFH$IGC')                  *
               ERASE MAPONLY WAIT RESP(RESP)
         CLC  RESP,DFHRESP(NORMAL)     CHECK FOR NORMAL RESPONSE
         BNE  ERROR1
 2)      EXEC CICS RECEIVE MAP('MAPA') MAPSET('DFH$IGC') RESP(RESP)
         CLI  EIBAID,DFHCLEAR          WAS CLEAR KEY PRESSED?
         BE   CLEAR                    ... YES, GO TO CLEAR
         CLC  RESP,DFHRESP(MAPFAIL)    HAS THE MAP BEEN READ IN OK?
         BE   MAPFAIL                  ... NO, GO TO MAPFAIL
         CLC  RESP,DFHRESP(NORMAL)     CHECK FOR NORMAL RESPONSE
         BNE  ERROR1
```

Figure 10 (Part 1 of 4). Sample 4: CICS-to-CICS conversation - combined front-end and back-end transaction (DFH$ICIC)

```
         LA    8,DATAI
         MVC   DATABL(3+L'DATABO),DATAL
         MVC   OUTLEN,DATAL
 3)      EXEC CICS SEND MAP('MAPB') MAPSET('DFH$IGC')                  *
         WAIT ERASE RESP(RESP)
         CLC  RESP,DFHRESP(NORMAL)
         BNE  ERROR1
 4)      EXEC CICS ALLOCATE SYSID(SYSIDI) RESP(RESP)
         CLC  RESP,DFHRESP(SYSIDERR)   IS THE SYSTEM ID VALID?
         BE   SYSERR                   ... NO, GO TO SYSERR
         CLC  RESP,DFHRESP(NORMAL)
         BNE  ERROR1
         MVC   ATCHSESS,EIBRSRCE
CONVERSE DS    0H
         MVC   INLEN,=H'2048'
 5)      EXEC CICS CONVERSE                                            *
               SESSION(ATCHSESS)                                       *
               FROM(0(8))                                              *
               FROMLENGTH(OUTLEN)                                      *
               INTO(REMDATA)                                           *
               TOLENGTH(INLEN)                                         *
               RESP(RESP)
         CLC  RESP,DFHRESP(NORMAL)
         BNE  ERROR1
         MVC   XDFEIFLG,EIBSYNC        SAVE EIB VALUES
DATASENT DS    0H
 6)      CLC   INLEN,=H'0'             IF NULL RU SENT
         BE    TESTSYNC                NOTHING TO SEND.
         LH    1,INLEN
         LA    2,REMDATA(1)            ADDR BYTE AFTER DATA
         MVI   0(2),X'13'              INSERT CURSOR HERE
         LA    1,1(,1)
         STH   1,INLEN
         EXEC CICS SEND TEXT FROM(REMDATA) LENGTH(INLEN)               *
               ACCUM RESP(RESP)
         CLC  RESP,DFHRESP(NORMAL)
         BNE  ERROR1
TESTSYNC DS    0H
 7)      CLI   XSYNC,X'FF'
         BNE   TESTFREE
         EXEC CICS SYNCPOINT RESP(RESP)
         CLC  RESP,DFHRESP(NORMAL)
         BNE  ERROR1
TESTFREE DS    0H
 8)      CLI   XFREE,X'FF'
         BNE   TESTRECV
         EXEC CICS SEND PAGE RETAIN RESP(RESP)
         CLC  RESP,DFHRESP(NORMAL)
         BNE  ERROR1
         EXEC CICS RETURN
```

Figure 10 (Part 2 of 4). Sample 4: CICS-to-CICS conversation - combined front-end and back-end transaction (DFH$ICIC)

```
TESTRECV DS    0H
 9)      CLI   XRECV,X'FF'
         BNE   SEND
         MVC   INLEN,=H'2048'
         EXEC CICS RECEIVE SESSION(ATCHSESS) INTO(REMDATA)            *
               LENGTH(INLEN) RESP(RESP)
         CLC  RESP,DFHRESP(NORMAL)
         BNE  ERROR1
         MVC   XDFEIFLG,EIBSYNC        SAVE EIB VALUES
         B     DATASENT
SEND     DS    0H
 10)     EXEC CICS SEND PAGE RETAIN RESP(RESP)
         CLC  RESP,DFHRESP(NORMAL)
         BNE  ERROR1
         MVC   OUTLEN,=H'60'
         EXEC CICS RECEIVE INTO(CONTROL) LENGTH(OUTLEN) RESP(RESP)
         CLC  RESP,DFHRESP(NORMAL)
         BNE  ERROR1
         LH    0,OUTLEN
         SH    0,=H'3'                 FOR LENGTH OF SBA
         LA    8,CDATA
         B     CONVERSE
*
ERROR1   DS    0H
         MVC   MESSAGE,ERRMSG
         B     EXPLAIN
ERRMSG   DC    CL32'ERROR - TRANSACTION TERMINATED'
*
SYSERR   DS    0H
 11)     CLI   EIBRCODE+1,12
         BE    UNKNOWN
         CLI   EIBRCODE+1,8
         BE    OUTSERV
         CLI   EIBRCODE+1,4
         BE    NOTCTSE
NOLINK   DS    0H
 12)     MVC   MESSAGE,LINKMSG
         MVC   MESSAGE+28(4),SYSIDI
         B     EXPLAIN
LINKMSG  DC    CL32'UNABLE TO ESTABLISH LINK TO       '
*
UNKNOWN  DS    0H
 13)     MVC   MESSAGE,UNKMSG
         MVC   MESSAGE+12(4),SYSIDI
         B     EXPLAIN
UNKMSG   DC    CL32'SYSTEM NAME      IS NOT KNOWN   '
```

Figure 10 (Part 3 of 4). Sample 4: CICS-to-CICS conversation - combined front-end and back-end transaction (DFH$ICIC)

```
OUTSERV  DS    0H
 14)     MVC   MESSAGE,OUTSVMSG
         MVC   MESSAGE+8(4),SYSIDI
         B     EXPLAIN
OUTSVMSG DC    CL32'LINK TO      IS OUT OF SERVICE'
*
NOTCTSE  DS    0H
 15)     MVC   MESSAGE,TCTMSG
         MVC   MESSAGE(4),SYSIDI
         B     EXPLAIN
TCTMSG   DC    CL32'     IS NOT A SYSTEM NAME'
*
EXPLAIN  DS    0H
         EXEC CICS SEND FROM(MESSAGE) LENGTH(=H'32')                    *
               ERASE WAIT RESP(RESP)
         CLC  RESP,DFHRESP(NORMAL)
         BNE  ERROR1
CLEAR    DS    0H
         EXEC CICS SEND CONTROL FREEKB RESP(RESP)
         CLC  RESP,DFHRESP(NORMAL)
         BNE  ERROR1
         EXEC CICS RETURN
         END
```

Figure 10 (Part 4 of 4). Sample 4: CICS-to-CICS conversation - combined front-end and back-end transaction (DFH$ICIC)

Program notes for DFH$ICIC

1. The screen is cleared, and the prompting map is displayed at the terminal.
2. The remote system name and command to be transmitted are mapped in. Branches are taken on error or “terminate” conditions.
3. The screen is cleared again and the command entered by the operator is displayed on the top line.
4. A session is now allocated naming the remote system only, and its name is obtained from EIBRSRCE.
5. A CONVERSE command is now issued that sends the data entered by the terminal operator to the specified remote system, then receives the resulting response from that system. To enable the program to determine what action is next expected of it, the contents of the EXEC interface block are examined; thus the values therein must be retained. The SESSION option is used because the application is requesting that an alternate facility be made available to it. Note that, although it is permissible to build an attach header and transmit it using the CONVERSE command, this action does not need to be taken in this case because CICS assumes by default that the first four characters of the transmitted data contain the transaction code.

6. If the data length field for the RECEIVE component of the CONVERSE command indicates that there is data to be handled, a logical message is built using the BMS TEXT facility for subsequent sending to the screen. To ensure that the terminal cursor is placed on the next available line for any further input, the “Insert Cursor” control character is appended to the data stream.
7. The session-oriented information transmitted across the LU6 session by the remote transaction must now be examined to determine what action should be taken next. The “SYNCPOINT required” indicator in the EXEC interface block is first tested and if need be the program issues its own SYNCPOINT command.
8. If the EXEC interface block (EIB) indicates that the program should now free the session, thereby denoting that the remote transaction has completed successfully and has terminated the conversation, the built logical message is sent to the screen using the RELEASE option of the SEND PAGE command, which returns control direct to CICS and thus frees the session.
9. If the EXEC interface block (EIB) indicates that the application is to continue receiving data from across the session, a further RECEIVE command is issued.
10. The indicators SYNCPOINT, FREE session, or RECEIVE do not apply, thus by default the remote application has requested a further transmission from this program. (In the case of the CICS-supplied programs named in the description above, this would imply the receipt of a prompting message.) The program therefore sends the logical message built to date, which includes the prompt, to the terminal operator and receives a reply; a second CONVERSE command can then be issued across the session. Note that the “Set Buffer Address” control and the two buffer address bytes received from the terminal must be bypassed before transmission across the link.
11. The SYSID error routine has been entered. To determine the exact cause of the error, EIBRCODE must be examined, and an appropriate information message sent to the operator.
12. Some kind of error exists that prevents the link between the two systems from being established.
13. The remote system name given by the operator is not recognized.
14. The link to the remote system is out of service.
15. The system name given is recognized, but is not that for a remote system.

Part 4. BMS partition and transient data samples

Chapter 17. The BMS partition samples 153

Chapter 18. The transient data sample (DFH$TDWT) 155

© Copyright IBM Corp. 1989,1999

Chapter 17. The BMS partition samples

This part of the book describes the BMS partition samples in COBOL and PL/I.

Overlapping operator keystrokes

Sample programs are provided in COBOL and PL/I which illustrate overlapped keystroking into two BMS partitions. The source programs are in the PRD1.BASE[2] library and are named DFH0CPKO for COBOL, and DFH$PPKO for PL/I. (Contact your system programmer for the name of this library as installed on your system.)

Look-aside querying

Sample programs are provided in COBOL and PL/I that show overlapped keystroking into one BMS partition while look-aside queries can be made using another BMS partition. The source code for these programs is in the CICS-supplied PRD1.BASE library[2] and is named DFH0CPLA for COBOL and DFH$PPLA for PL/I.

Installing the sample group

Install the group using the CEDA INSTALL GROUP (DFH$BMSP) command.

Table 8. Sample program

Language	Program name	Map names in program	Source map names	Partitionset
COBOL	DFH0CPKO DFH0CPLA	DFH0CGP	DFH0CMP	DFH0PS
PL/I	DFH$PPKO DFH$PPLA	DFH$PGP	DFH¢PMP	DFH$PS

Invoking the sample group

You can invoke the sample group by entering the transaction identifier on a clean screen.

Table 9. Sample group and its identifier

Language	TRANSACTION	
	Keystroke overlap	Lookaside query
COBOL	XPKO	XPLA
PL/I	PPKO	PPLA

[2] This is the name of the library as supplied by IBM. Your installation may be using another name. Check with your systems programmer

© Copyright IBM Corp. 1989,1999

Chapter 18. The transient data sample (DFH$TDWT)

CICS provides a sample which prints messages on a local 3270 printer as they occur. In the destination control table (DCT), the user can specify that messages such as those from the abnormal condition program (DFHACP) and sign-on and sign-off messages, should be sent to destinations defined in the DCT with TYPE=INDIRECT. If these destinations are defined (by means of INDDEST) so that they refer to an intrapartition destination with a transaction identifier and a trigger level of 1, the receipt of a message will cause that transaction to be started.

The transaction will invoke the DFH$TDWT sample program, which prints the message at a local terminal.

To use this sample, the CICS system must include automatic transaction initiation and an intrapartition transient data set. The source code is provided in CICS170.SAMPLIB, and the object code is provided in CICS170.LOADLIB.

Sample table entries for this technique are as follows:

```
DFHDCT    TYPE=INDIRECT,DESTID=CSCS,INDDEST=LPRT
DFHDCT    TYPE=INDIRECT,DESTID=CSTL,INDDEST=LPRT
DFHDCT    TYPE=INDIRECT,DESTID=CSML,INDDEST=LPRT

DFHDCT    TYPE=INTRA,         TO AUTO INIT TASK
          DESTID=LPRT,        LOCAL 3270 PRINTER
          TRIGLEV=1,
          TRANSID=TDWT,
          DESTFAC=TERM

DFHPCT    TYPE=ENTRY,
          TRANSID=TDWT,
          PROGRAM=DFH$TDWT

DFHPPT    TYPE=ENTRY,
          PROGRAM=DFH$TDWT

DFHTCT    TYPE=TERMINAL,
          TRMIDNT=LPRT,
          TRMTYPE=L3286
```

Note: The DESTID in the DCT TYPE=INTRA macro instruction and the TRMIDNT in the TCT TYPE=TERMINAL macro instruction must be the same.

© Copyright IBM Corp. 1989,1999

Bibliography

CICS Transaction Server for VSE/ESA Release 1 library

Evaluation and planning	
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
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
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 CICS Interface	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

© Copyright IBM Corp. 1989,1999

Books from VSE/ESA 2.4 base program libraries

VSE/ESA Version 2 Release 4

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 2.4 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.

© Copyright IBM Corp. 1989,1999

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:

C/370	CICS
CICS OS/2	
Common User Access	CUA
IBM	
VTAM	

Index

A

action lists 18
application action bar 5, 16, 19, 29
  content 20
  fields 19
  interaction 21
  keyboard interaction 21
  layout 20
  pull-down window 21
  selected emphasis 21
  using 19
assembler language
  FILEA sample application 125
audio interaction 24

B

base map definition 22
basic mapping support definition 8
  minimum, standard, full 8
BMS maps
  CUA text model BMS maps 101
  FILEA sample BMS maps 133
BMS support 5
BMS/CUA color differences 13

C

C
  FILEA sample application 125
canned map definition 22
COBOL
  CUA text model 67
  FILEA sample application 125
COBOL/VSE 1
color and emphasis 13
color assumptions 13
column headings and group headings 12
command area 24
common errors 101
common user access definition 6
copyright information 24
CUA consistency 6
  physical consistency 6
  semantical 6
  syntactical consistency 6
CUA text model overview
  application action bar use 49
  browse from an action list 40
  browse panel 37
  contextual help 45
  designer's view 47
CUA text model overview *(continued)*
  file I/O 47
  file pull-down 34
  help panels 47
  help pull-down 44
  help stub 44
  list panel 39
  open pop-up 38
  pop-up windows 49
  print stub 46
  program structure 50
  pull-down window 49
  resource usage 48
  routing 51
  saving a customer record 43
  starting out panel 33
  update after verification 42
  update from an action list 41
cursor position 15

D

demonstration application 47
descriptive text 12
distributed transaction processing sample applications 2, 135

E

entry field appearance 17
entry fields 17

F

field prompts 12
file-open pop-up 35
FILEA sample programs
  browse (assembler language) 126
  browse (C) 126
  browse (COBOL) 126
  browse (PL/I) 126
  inquiry/update (assembler language) 126
  inquiry/update (C) 126
  inquiry/update (COBOL) 126
  inquiry/update (PL/I) 126
  low balance report (assembler language) 129
  low balance report (C) 129
  low balance report (COBOL) 129
  low balance report (PL/I) 129
  operator instruction (assembler language) 125
  operator instruction (C) 125
  operator instruction (COBOL) 125
  operator instruction (PL/I) 125

© Copyright IBM Corp. 1989,1999

FILEA sample programs *(continued)*
order entry (assembler) 127
order entry (C) 127
order entry (COBOL) 127
order entry (PL/I) 127
order entry queue print (assembler language) 128
order entry queue print (C) 128
order entry queue print (COBOL) 128
order entry queue print (PL/I) 128
function key area 25
keys supported 25

H

help stub 44

I

installation
CUA text model installation 55
FILEA sample installation 121
instructions 12
intercommunication facilities sample applications 2, 135

K

keystroke overlapping 153

L

language considerations 122
list processing 30
look-aside query 153

M

messages 101
action messages 23
message area 23
message content 24
message location 23
message removal 24
notification messages 23
types of messages 23
warning messages 23
multi-line commands 24

N

national language 15
nonprogrammable workstation (NWS) 6

O

overlapping keystroking 153
overlay map definition 22
overlaying maps 101
cursor position 101

P

panel area separators 12
panel elements 11
panel id 11
panel title 11
partition samples 153
personal computers 5
PL/I
FILEA sample application 125
pop-ups 101
pregenerated system
sample VSAM file 131
print 46
programmable workstation (PWS) 6
programs, FILEA sample (assembler language) 125
programs, FILEA sample (C) 125
programs, FILEA sample (COBOL) 125
programs, FILEA sample (PL/I) 125
prompt 19
protected text 12
pseudo-conversational 1
pull-down window 19, 21
using 19
pull-downs 101
pull-downs and pop-ups 22
BMS support 22
path length 22

R

retrieve 24
rules to work by 3

S

SAA (Systems Application Architecture) 5
sample application programs
browse FILEA sample program (3270) 126
distributed transaction processing sample applications 2, 135
FILEA sample applications 119
operator instruction FILEA sample program (3270) 125
order entry FILEA sample program (3270) 127
order entry queue print FILEA sample program (3270) 128
report FILEA sample program (3270) 129
update FILEA sample program (3270) 126
sample utilities
intercommunication facilities 2, 135

scrollable entry fields 17
scrollable selection fields and lists 17
scrolling panel areas 25
selection cursor, moving 15
selection element emphasis 17
selection field initial conditions 17
selection fields 16
Systems Application Architecture (SAA) 5

T

threshold 9, 29
  threshold 1 9
  threshold 2 10
  threshold 3 10
  threshold 4 10
transient data sample 155

U

unavailable emphasis 17
user interface design 9

W

window sizing 11

Sending your comments to IBM

CICS® Transaction Server for VSE/ESA™

Sample Applications Guide

SC33-1713-00

If you want to send to IBM any comments you have about this book, please use one of the methods listed below. Feel free to comment on anything you regard as a specific error or omission in the subject matter, and on the clarity, organization or completeness of the book itself.

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:

  IBM UK Laboratories
  Information Development
  Mail Point 095
  Hursley Park
  Winchester, SO21 2JN
  England

- By fax:
  - From outside the U.K., after your international access code use 44 1962 870229
  - From within the U.K., use 01962 870229
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Email: idrcf@hursley.ibm.com

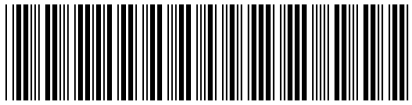
Whichever method you use, ensure that you include:

- 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.

Program Number: 5648-054

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

SC33-1713-00



Spine information:

CICS TS for VSE/ESA Sample Applications Guide *Release 1*