MVS to AIX Application Migration Cookbook

Document Number GG24-4375-00

August 1994

International Technical Support Organization
San Jose Center

Take Note!

Before using this information and the products it supports, be sure to read the general information under "Special Notices" on page xvii.

First Edition (August 1994)

This edition applies to Version 1, Release 1 of the IBM licensed program AIX Customer Information Control System/6000 (CICS/6000), program number 5765-148; Version 1, Release 1, Modification Level 0, of the IBM licensed program DATABASE 2 AIX/6000* product, program number 5765-172 for use with Version 3, Release 2, Modification Level 5, of the AIX/6000 Operating System; Version 3, Release 3, Modification Level 0 of the IBM licensed program Customer Information Control System/Enterprise Systems Architecture (CICS/ESA), program number 5685-083; and Version 3, Release 1, Modification 0 of the IBM licensed program DATABASE 2 (DB2), program number 5685-DB2.

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

An ITSO Technical Bulletin Evaluation Form for reader′s feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 471 Building 70B
5600 Cottle Road
San Jose, CA 95193

Abstract

This document provides information on the steps required to migrate a Customer Information Control System (CICS) and DATABASE 2 (DB2) application in a Multiple Virtual System (MVS) environment to an Advanced Interactive Executive/6000 (AIX) environment using CICS/6000 and DB2/6000. It focuses on COBOL applications but is kept on a general level where possible.

This document is written for technical professionals assisting in a migration project of this kind. Some knowledge of MVS, AIX, CICS, and DB2 is assumed. Knowledge of Distributed Computing Environment (DCE) and Encina is required.

Configuration and setup information is provided where considered helpful. The SNA/Server 6000 profiles used for connecting the CICS MVS system with the CICS/6000 system are provided as samples.

AD (173 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help application programmers, system programmers, and system engineers involved in a downsizing project to accomplish all necessary tasks, including data transfer, database migration, program recompilation, and map generation, to migrate an application from a mainframe system to an IBM RISC System/6000 running the AIX/6000 operating system. The information in this publication is not intended as the specification of any programming interfaces that are provided by IBM Customer Information Control System/6000 (CICS/6000) or by IBM DATABASE 2 AIX/6000. See the PUBLICATIONS section of the IBM Programming Announcement for IBM Customer Information Control System/6000 (CICS/6000) and IBM DATABASE 2 AIX/6000 for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 208 Harbor Drive, Stamford, CT 06904 USA.

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

The following document contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples contain the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Reference to PTF numbers that have not been released through the normal distribution process does not imply general availability. The purpose of including these reference numbers is to alert IBM customers to specific information relative to the implementation of the PTF when it becomes available to each customer according to the normal IBM PTF distribution process.

The following terms, which are denoted by an asterisk (*) in this publication, are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AIX	AIX/6000
AIXwindows	CICS
CICS/ESA	CICS/6000
Common User Access	CUA
DATABASE 2	DB2
DB2/6000	Distributed Relational Database Architecture
DRDA	GDDM
IBM	IMS/ESA
MQSeries	MVS/ESA
NetView	OPC
OS/2	QMF
RISC System/6000	SAA
Systems Application Architecture	VM/ESA
VTAM	

The following terms, which are denoted by a double asterisk (**) in this publication, are trademarks of other companies:

BACHMAN/DBA Catalog Extract for DB2	Bachman Information Systems, Inc.
BRIDGE/FASTLOAD for DB2/6000	Bridge Technology Inc.
DCE, Distributed Computing Environment	Open Software Foundation
Encina	Transarc Corporation
HIREL	SWS Software GmbH
Informix	Informix Corporation
Oracle	Oracle Corporation
Sybase	Sybase, Inc.
X/Open	X/Open Company Limited

Other trademarks are trademarks of their respective companies.

Preface

This document is intended to be a guideline for use before and during a downsizing project. It contains a list of all actions you will have to take to downsize an application, as well as hints, tips, and conclusions from the experience gained in a successfully completed downsizing project.

This document is intended for application programmers, system programmers, system engineers, and any other participants in a downsizing project.

How This Document Is Organized

The document is organized as follows:

- Chapter 1, "Introduction" presents the definition of downsizing applied to this project and describes how you should approach a downsizing project.
- Chapter 2, "Objectives and Scope" states the objectives of this book and of the project that led to this book.
- Chapter 3, "Recommendations" offers some guidelines and recommendations for downsizing projects.
- Chapter 4, "Configurations" shows possible migration paths and explains how we configured the systems and the network for our downsizing project.
- Chapter 5, "Transferring Sequential Files" describes the different ways of transferring sequential files from the MVS mainframe to the AIX system.
- Chapter 6, "Converting and Transferring DB2 Databases" describes the different ways of transferring your DB2 data from DB2 MVS to DB2/6000.
- Chapter 7, "Migrating DB2 Objects" compares the DB2 objects found in DB2 MVS with those found in DB2/6000. It also explains the changes necessary to convert the SQL create statements for DB2 MVS to DB2/6000.
- Chapter 8, "Migrating Maps" outlines the steps to prepare CICS BMS maps for CICS/6000.
- Chapter 9, "Migrating COBOL Programs" outlines the steps to convert and compile COBOL programs using both embedded SQL and the CICS API on the AIX system.
- Chapter 10, "Terminal Access to Your CICS/6000 Region" describes the definitions and changes required to access CICS/6000 from your TCP/IP network.
- Chapter 11, "Batch Jobs" provides an overview of the available job scheduling systems for AIX and explains how to invoke batch jobs on AIX.
- Chapter 12, "Transferring VSAM Files" describes how to transfer VSAM data to your AIX system and how to load it into SFS.
- Chapter 13, "Printing" shows you how to issue print jobs from CICS/6000.
- Appendix A, "Migration Tidbits" provides you with information about third-party tools, IBM services, and additional information sources.
- Appendix B, "Integrating CICS/6000 and DB2/6000" explains how to set up the XA interface between CICS/6000 and DB2/6000.

- Appendix C, “MVS Definitions” lists all of the definitions we made on the MVS system for our downsizing project.
- Appendix D, “AIX Definitions” lists all the definitions we made on the AIX system for our downsizing project.
- Appendix E, “Overcoming Full Function BMS Restrictions” explains how to migrate your full function BMS programs by coding paging emulation transactions.
- Appendix F, “JCL Conversion” explains how to convert JCL to AIX shell script.
- Appendix G, “EBCDIC to ASCII Conversion” contains a sample program for partially converting files from EBCDIC to ASCII.
- Appendix H, “Interfacing COBOL with Encina SFS Files” shows you how to use the external file handler for accessing SFS files from a batch program.

Related Publications

The following publications are considered particularly suitable for a more detailed discussion of the topics covered in this document:

- *DB2 V3 Administration Guide*,(3 volumes), SC26-4888
- *DB2 V3 Application Programming and SQL Guide V3R3*, SC26-4889
- *DB2 V3 SQL Reference*, SC26-4890
- *CICS/ESA Application Programming Guide V3R3*, SC33-0675
- *CICS/ESA Application Programming Reference V3R3*, SC33-0676
- *CICS/ESA System Programming Reference V3R3*, SC33-0670
- *CICS/ESA Intercommunication Guide V3R3*, SC33-0657
- *CICS/ESA Resource Definition(Online) V3R3*, SC33-0666
- *CICS/ESA Resource Definition(Macro) V3R3*, SC33-0667
- *CICS/ESA CICS-Supplied Transactions V3R3*, SC33-0669
- *CICS/ESA Sample Applications Guide V3R3*, SC33-0731
- *CICS Family: API Structure*, SC33-1007
- *AIX CICS/6000 CICS-Supplied Transactions*, SC33-0813
- *AIX CICS/6000 Application Programming Guide*, SC33-0814
- *AIX CICS/6000 Intercommunication*, SC33-0815
- *AIX CICS/6000 Planning and Installation*, SC33-0816
- *AIX CICS/6000 Application Programming Reference*, SC33-0886
- *AIX CICS/6000 Customization and Operation*, SC33-0931
- *DATABASE 2 AIX/6000 Installation Guide*, GC09-1570
- *DATABASE 2 AIX/6000 Administration Guide*, SC09-1571
- *DATABASE 2 AIX/6000 Programming Guide*, SC09-1572
- *DATABASE 2 AIX/6000 Programming Reference*, SC09-1573
- *DATABASE 2 AIX/6000 and DATABASE 2 OS/2 SQL Reference*, SC09-1574
- *DATABASE 2 AIX/6000 Command Reference*, SC09-1575

- *IBM AIX Version 3.2 for RISC System/6000 Commands Reference (four volumes)*, GBOF-1802
- *Formal Register of Extensions and Differences in SQL*, SC26-3316
- *BRIDGE/FASTLOAD for DB2/6000 Server Edition Operations Manual*, Bridge Technology Inc.
- *BRIDGE/FASTLOAD for DB2/6000 Mainframe Edition Operations Manual*, Bridge Technology Inc.

International Technical Support Organization Publications

- *AIX CICS/6000 and RDBMSs Integration: Experiences with the XA Interface*, GG24-4214 (available 3Q 1994)
- *AIX CICS/6000 Installation and Configuration: A Guide to Implementation*, GG24-4091
- *A Guided Tour of SNA Server/6000 Version 2.1*, GG24-4189
- *Rightsizing the Database Environment: DB2 for MVS to DB2/6000*, GG24-4431-00 (available 4Q 1994)

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

Bibliography of International Technical Support Organization Technical Bulletins, GG24-3070.

To get listings of ITSO technical bulletins (redbooks) online, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order ITSO Technical Bulletins (Redbooks)

IBM employees in the USA may order ITSO books and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their IBM branch office.

Customers may order hardcopy redbooks individually or in customized sets, called GBOFs, which relate to specific functions of interest. IBM employees and customers may also order redbooks in online format on CD-ROM collections, which contain the redbooks for multiple products.

Acknowledgments

This publication is the result of a residency conducted at the International Technical Support Organization, San Jose Center.

The authors of this document are:

Hennie Kok
ISM South Africa

Arturo Algueiro Melo
IBM Argentina

Martin Prater
IBM Switzerland

Thanks to the following people for the invaluable advice and guidance provided in the production of this document:

Guido De Simoni
International Technical Support Organization, San Jose Center

Maggie Cutler
Editor

Dan Gibson
IBM Atlanta

Bill Karras
International Technical Support Organization, Poughkeepsie

Roger Packham
IBM Switzerland

Howard Rowe
IBM UK

Dennis Weiand
IBM Dallas

Herbert Wildhaber
IBM Switzerland

The advisor for this project was:

Joe DeCarlo
International Technical Support Organization, San Jose Center

IBM COBOL Goes Desktop

In October 1994, IBM intends to announce its new workstation-based COBOL products. With this introduction, you can move your IBM COBOL applications into the world of object-oriented, client-server, and visual graphical user interfaces (GUIs)—on multiple platforms.

The new IBM Visual COBOL for OS/2 and IBM COBOL for AIX and enhancements to the existing IBM COBOL for MVS and VM provide multiple platform and client-server support: taking the premier IBM COBOL product to the workstation with a rich set of COBOL productivity tools. These COBOL products support local and remote data access through distributed DB2, CICS, IMS Client Server/2, and VSAM.

System Object Model (SOM)-based object-oriented extensions have been added to IBM COBOL for MVS and the new IBM Visual COBOL for OS/2, providing consistent object-oriented language support across those platforms. To enhance your application development of object-oriented programs, access to existing SOM-based class libraries is supported.

The IBM COBOL family has been extended on OS/2 with an integrated set of tools to support visual development of GUIs for the OS/2 platform and development of multiplatform client-server code from an OS/2 workstation. Tools include a COBOL-language-sensitive editor and debugger, a visual GUI builder, a transaction assistant to help invoke existing CICS transactions, and a data assistant to help generate SQL queries and COBOL data structures.

Chapter 1. Introduction

In this chapter we present the definition of downsizing that we applied to our project and discuss incorporating downsizing into your information technology (IT) strategy. Additionally we discuss the recommended approach to your first downsizing project, looking at what you can safely attempt to accomplish and what you should split into multiple projects. At the end of this chapter you can find some hints on how to choose the right application for your first downsizing project.

1.1 What Is Downsizing?

Before you attempt any downsizing project you and your organization must have a clear concept of downsizing, as well as realistic expectations of the results and benefits that can be achieved.

The definition of downsizing we applied to this project is defined in Figure 1.

Migrating a fully functional application from a mainframe environment to a workstation.

Figure 1. Definition of Downsizing

It is important to recognize the difference between downsizing and exploiting a client/server environment. In downsizing the emphasis is on maintaining the existing function using a compact infrastructure that offers flexibility in placing the computing resources where they are used. In contrast client/server computing focuses on the distribution of the presentation, application logic, and data access layers across multiple platforms.

We approached our downsizing project from the point of view of an application programmer. Where possible we discuss some system or network management aspects, but the focus is on business applications.

1.2 Downsizing As Part of Your Information Technology Strategy

One must not underestimate the importance of making a decision in favor of downsizing. To achieve the desired results it is necessary to plan this step carefully. In addition, the IT strategy should reflect a commitment to downsizing to guarantee that the required hardware and software infrastructure as well as skills are available before the project starts.

You will have to not only provide the resources for the downsizing project but also maintain and operate the new environment. Typically, the maintenance and operation of the new environment require a new set of skills. Workstations, as their name implies, are located near users; they are not under the control of a central information systems group. Such physical distribution raises new issues when planning for security. The classical *closed shop* computing centers cannot be implemented the way they used to be for a physically distributed environment.

The need for management tools for the new distributed environment has been recognized. Various tools are available to aid you in software distribution, network management, and distributed system management. The Open Systems Foundation (OSF) is working on a standard for the Distributed Management Environment (DME). IBM* already offers you distributed management tools in the form of NetView/6000* for managing your network, NetView/Distribution Manager* for keeping your software up to date, and Distributed System Management Interface Tool (DSMIT) for managing the system environments of decentralized workstations from one central workstation.

If your organization is taking its first steps in the workstation direction we recommend that you define a Transmission Control Protocol/Internet Protocol (TCP/IP) network strategy. You should decide on how to incorporate this new network into your existing network. Another important decision to make is whether you want to be connected to the official Internet. Being connected to the official Internet influences the IP addresses that you can use in your network. Customer Information Control System/6000 (CICS/6000)* uses the Distributed Computing Environment (DCE)**. TCP/IP is a prerequisite for DCE and must be installed and configured before CICS/6000 can use DCE. However, you cannot change some parts of your TCP/IP configuration, namely the IP addresses, without reconfiguring DCE, and reconfiguring DCE impacts the operation of your CICS/6000 and Encina** environments. Therefore we recommend that you have the definitive TCP/IP configuration and addresses available before you install DCE, Encina, and CICS/6000.

DCE offers you a multitude of configuration possibilities. An important aspect to consider is how to define your DCE cells. You should consult the available documentation on DCE cell design before you decide on how to set up your cells. The distribution of the various components of DCE but also of Encina and CICS/6000 will influence the design.

1.3 One Step At a Time

Often new platforms offer a variety of new options for the application programmer to use. Planning for too many changes in a new environment can increase the overall complexity of a downsizing project and endanger a successful completion. Through the experience gathered in this project we recommend that you migrate one application at a time and maintain the function provided on the mainframe. Once you have migrated the application you can revamp it by providing a graphical user interface (GUI) or implementing a client/server architecture.

We believe that mainframe application programmers will feel comfortable using the AIX/6000* versions of CICS*, DATABASE 2 (DB2)*, and COBOL. They will easily be able to do the maintenance for the downsized application. If you want to use an AIX-specific product like AIXwindows* or DCE, you will have to plan to train your programmers. AIXwindows programmers must have in-depth knowledge of GUIs in general, but predominantly of the Xlib, Xintrinsics, and Motif libraries. They should also be aware of the IBM Systems Application Architecture (SAA)* Common User Access (CUA)* standards, as well as any of your home-grown standards. The IBM AIX Interface Composer (AIC)* or other comparable computer-aided software engineering (CASE) tools can help you grow quickly into AIXwindows programming and increase your productivity. They cannot, however, replace formal training.

Migrating one application at a time will show you which application development infrastructure is available on the AIX* platform. A lot of this infrastructure is contained in the AIX base system. Other more sophisticated products, for example, for version control, can be found in the AIX application development line of products. You may also have self-developed products for your mainframe application development that you would like to migrate. The one step at a time approach allows you to gather preliminary experiences in the AIX application development field and helps you choose and introduce the appropriate infrastructure products for your application development.

1.4 Selecting the Right Applications

Downsizing offers the opportunity to protect your investment in application programs while enabling you to benefit from new technologies. In some cases you may want to rewrite the application specifically for the new platform rather than migrate the existing code.

Before choosing an application for downsizing, especially if it is your first downsizing project, consider the following factors:

- Number of users
- Number of programs and/or transactions
- Amount of data
- Complexity of the application
- Familiarity with the application.

The objectives of your downsizing project will influence your choice of application. On the one hand, if the main purpose of the project is to test the feasibility of downsizing, you should not choose a very small and unimportant application. The results will not be representative of your larger core business applications. On the other hand, we do not recommend that you begin by downsizing your largest core business application. You should definitely assess the risks involved in downsizing. An ideal downsizing candidate is an application whose use of CICS, DB2, and COBOL facilities is representative and that has an acceptable volume of data and transactions and impacts the fewest users.

Chapter 2. Objectives and Scope

The environment for our project was an MVS/ESA*, DB2, and CICS mainframe system using COBOL and an IBM RISC System/6000* as the target system for downsizing. Our objective was twofold: to retain the investment in the current terminals and network and to investigate a workstation-only environment without any mainframe support.

To achieve our objectives, we developed the following scenarios:

- Scenario 1: Migrate the DB2 database, COBOL application, and CICS transaction processing to AIX while retaining the current terminals and network for CICS terminal access through the transaction routing function of CICS intersystem communication (ISC) between the MVS mainframe and AIX platform.
- Scenario 2: Migrate the DB2 database, COBOL application, and CICS transaction processing to an all-AIX environment without any support from the mainframe.

In this chapter we describe the objectives of this book, the areas we cover, as well as the audience that we address.

2.1 What You Can Expect

In this book we demonstrate the feasibilty of migrating a fully functional and operational MVS mainframe application written in COBOL, using CICS as the transaction manager and DB2 as the database. Naturally some of the issues discussed in this book will also apply to slightly different environments. Where possible we keep the discussion on a general level.

After reading this book you should be able to estimate the effort required to complete a similar downsizing project. You should also be able to develop a fairly accurate sense of the level of compatibility you can expect during migration. We hope to provide you with enough information to successfully complete your downsizing projects without having to encounter major pitfalls.

Please keep in mind that this project used various products of which some have only recently been released. The information provided in this book may be outdated by future releases of products. You should consult the documentation of the current release of your software products for definitive answers regarding features or restrictions.

You are advised to recognize the exploratory nature of this project. Our aim was to gather preliminary experiences in downsizing. It was out of the scope of this project to consider all aspects of a downsizing project. In our documentation we concentrate on the aspects proven important by the application we downsized.

2.2 What You Cannot Expect

Throughout this book we generally assume that you have successfully completed the installation and configuration of all required components. Nevertheless in certain situations we have chosen to provide you with basic configuration and setup information in areas that we consider useful and helpful. Please refer to the documents listed in the preface for detailed information about installation, configuration, and setup for all of the software components referenced in this book.

So as not to overload this book we have refrained from pursuing all issues not directly related to the application development focus of downsizing. Information on security, performance, operation, and management of a downsized system can be obtained from the documents listed in the preface.

You will find various references to IBM and non-IBM software and tools. Often we just mention that these tools and software exist, as we consider that valuable information. We were not able to test and review all of those tools and software to verify that they produce the promised results.

2.3 Audience

As an application developer involved in the migration of your application from a mainframe environment to an IBM RISC System/6000 workstation environment you will benefit most from this book. Although the focus is on the transfer of all of the application′s components and the rebuilding of the application, it is necessary to have a certain knowledge of the AIX and the MVS operating systems. A detailed knowledge of CICS, DB2, and COBOL is required to complete a downsizing project following the advice in this book.

Many of the details in this book could prove helpful for MVS and AIX system programmers or administrators who need to advise application developers on choosing the best program or process, for example.

This book contains useful information for anybody involved in a downsizing project, although it should be kept in mind that it is written for a technical audience.

2.4 How to Use This Book

You will gain the greatest benefit for your downsizing project if you read this book before you start your project. The book will help you estimate the effort and cost required to complete such a project. It also provides you with guidelines for choosing a suitable application for downsizing.

You will also find some useful information about setting up your AIX system together with some hints on configuration. You can use this book as a guideline during installation and configuration.

When you are implementing the downsizing project you can use this book as a handbook. A lot of the actions you have to perform can be done as described in this book without modification. In most cases all you have to do is select the best or appropriate solution for your environment and follow the steps outlined in this book.

2.5 Where to Go from Here

This book is intended to be used as a guide before starting and during a downsizing project. You will find a lot of information on which tasks you need to accomplish and which software you need to install and configure, but you will not find detailed information on the software products involved. Please refer to the list of publications in the preface for detailed information about the software products and tools you intend to use in your project.

In Appendix A, “Migration Tidbits” on page 97 you will find some useful information that can further aid you in your project. The appendix contains a list of available migration tools, other sources of information, and IBM service offerings. It is by no means complete, but it gives you an overview of the help you can expect to get from service providers and software tools. It also shows you where you can learn and ask questions about other tools.

Chapter 3. Recommendations

You can choose various configurations and setups for your environment. In this chapter we tell you why we recommend using the configuration and setup we have chosen. We also suggest the skills required for a downsizing project.

3.1 Set up a Link to the Mainframe

Setting up an Systems Network Architecture (SNA) link between your MVS system and IBM RISC System/6000 is the first and crucial step before starting your project. The link will enable you to use a 3270 terminal emulation program for online sessions as well as file transfer, if you do not have a TCP/IP connection available.

The SNA link will also enable you to use all of the CICS ISC functions—transaction routing, function shipping, and distributed program link. Transaction routing will enable you to protect your investment in your existing network and 3270 terminals. Function shipping and distributed program link may well be the most efficient methods of transferring VSAM data to the Encina Structured File System (SFS). Using the ISC functions, you can even choose to keep parts of your application on the mainframe (for example, VSAM data, assembly language routines).

You may also choose to implement a Distributed Database Connection Services (DDCS) connection between DB2/6000* and your mainframe DB2 system. This connection will enable you to export and import data from one system to another.

With all of the base products installed, setting up the link to mainframe as well as the ISC link and the DDCS connection requires about one personweek of effort. The experience gained in this project shows that this investment is well worthwhile.

3.2 XA Interface between DB2/6000 and CICS/6000

To access DB2/6000 from your CICS/6000 applications you can use either the XA interface or the non-XA interface. The XA interface is a definition of how a transaction manager (TM) communicates with a resource manager (RM). This definition is part of the X/Open* model for distributed transaction processing (DTP). In particular the XA interface provides support for the two-phase commit (2PC), which allows an application program to make updates coordinated by multiple RMs and still maintain integrity. The TM ensures that either all of the RMs or none of them commits the updates.

If you use the non-XA interface you have no integrity between the updates of your application program (transaction) in DB2/6000 and any other RMs. Even if DB2/6000 is the only database you plan to use, you are still using CICS/6000 and Encina resources for your files and queues. For this reason we recommend that you configure your CICS/6000 region to use the XA interface to communicate with DB2/6000.

In our case CICS/6000 and Encina are both TM and RM, as they coordinate commitment in all RMs but also manage resources such as files and queues.

DB2/6000 is another RM. Appendix B, “Integrating CICS/6000 and DB2/6000” on page 103 contains information about this interface. *AIX CICS/6000 and RDBMSs Integration: Experiences with the XA Interface* provides you with detailed background information on the XA interface. It also explains how to set up the interface to other databases such as Informix**, Oracle**, and Sybase**.

3.3 Required Skills

Ideally the project team for the downsizing project will combine good application and application development skills with knowledge of both the MVS and AIX environments. As this is not always possible, you should decide on which platform you will make any necessary changes to the source code, Basic Mapping Support (BMS) maps, or Structured Query Language (SQL) statements. Additional effort may be required if the team members have to use tools to which they are not accustomed, such as the editor or a script language. For the editing process, you can accommodate those users inexperienced in the AIX environment by installing tools available in both environments. For example, instead of the commonly used vi editor, which has a rather steep learning curve, you could use the e3 or the ISPF/PDF clone editor. Similarly for people new to the AIX environment REXX for AIX is easier to use than the standard Korn shell scripts.

For the installation and management of your system environment you will need good DCE and Encina skills. The application developer using CICS is only marginally confronted with DCE and with Encina, but it is useful to have a introductory knowledge of these components as an application developer.

3.4 Document Changes

You probably have to make changes to your source code for the downsizing project. You should document all of these changes very well. This will help you in future downsizing projects. It also helps you keep track of the changes made, so that you can reverse changes that did not work or that are no longer necessary in future releases of products. Ideally, you can put all of your changes into script files. In this way you can run the changes over your source code components again if necessary. The AIX stream editor, sed, provides a powerful script language for performing mass changes.

Chapter 4. Configurations

In this chapter we present the options you have for migrating your host applications to an AIX environment and outline the network configuration we used in our project. The definitions required for our setup for the MVS system can be found in Appendix C, "MVS Definitions" on page 107. Appendix D, "AIX Definitions" on page 111 contains all of the definitions for the AIX system.

4.1 Migration Approaches

There are several approaches to migrating MVS, CICS, and DB2 applications from a host-only environment to a totally distributed AIX, CICS/6000, and DB2/6000 environment. The approach you use depends on the requirements that determined your decision to downsize.

4.1.1 DB2/6000 Remote Database

The easiest way to begin the downsizing project is to migrate the relational database environment to AIX and establish a Distributed Relational Database Architecture (DRDA)* connection between DB2 in MVS/ESA and DB2/6000 in AIX (see Figure 2). You only have to define the host DB2 as an application in VTAM*, activate the distributed data facility (DDF) option in DB2, establish the appropriate SNA link between VTAM in MVS and SNA in the RISC System/6000, and include DB2/6000 in the AIX SNA definitions. As the host DB2 is a requester, you will have to define the DB2/6000 system in the host communications database.

The feature discussed here is DB2/6000 as an application server. It will be available with the announced version 2.1 of DB2/6000.

This link allows you to transfer the relational definitions and data by means of interactive DB2 statements. You do not have to modify the applications to access the new location.

```
+----------------------+                    +---------------+
|        MVS/ESA       |                    |      AIX      |
+-----------+----------+                    +---------------+
|           |          |                    |               |
|           |          ===========/         |   DB2/6000    |
| CICS/ESA  |    DB2   |         /          |               |
|           =    DDF   |        /=============  Remote  DB  |
|           |          |                    |    Server     |
|           |          |                    |               |
|           |          |                    |               |
+-----------+----------+                    +---------------+
```

Figure 2. DB2/6000 Remote Database

4.1.2 Remote CICS/6000 Application Owning Region (Scenario 1)

If your intent is to retain your investment in the terminals and network, while relieving the host system of processing overhead, the solution is to run the applications in the AIX environment and use the host CICS as a remote terminal owning region (TOR), connected through ISC with the CICS/6000 that runs in the AIX as an application owning region (AOR). This is the schema used in our Scenario 1, as shown in Figure 3.

You will have to establish the appropriate SNA link between VTAM in MVS and SNA in the RISC System/6000, then include CICS/6000 in the AIX SNA, DCE, and Encina definitions, and define the ISC sessions in the host CICS. You may refer to the definitions we used for our project in Appendix C, “MVS Definitions” on page 107 and Appendix D, “AIX Definitions” on page 111.

You also have to migrate the application programs and maps and recompile them, making any necessary adjustments. You can access the applications by defining the transactions as remote on the host side or using the CICS-supplied transaction, CRTE.

```
+---------------+                    +------------------------+
|   MVS/ESA     |                    |  RISC SYSTEM/6000 AIX  |
+---------------+                    +-------------+----------+
|               |                    |             |          |
| CICS/ESA TOR  ===========/         |             |          |
|               |         /          |  CICS/6000  | DB2/6000 |
|               |        /============= Remote  AOR =         |
|               |                    |             |          |
+---------------+                    +-------------+----------+
```

Figure 3. Scenario 1: Remote CICS/6000 AOR

4.1.3 Remote CICS/6000 AOR with DRDA Database

If the amount of data or the complexity of the relational model makes impractical the migration of the database to AIX as in scenario 1, while retaining the application running in AIX, the solution is to establish a DRDA connection back to the host, as shown in Figure 4 on page 13. This option combines the two discussed before, in 4.1.1, “DB2/6000 Remote Database” on page 11 and 4.1.2, “Remote CICS/6000 Application Owning Region (Scenario 1).” The only difference is that in this case, the host DB2 is a server instead of a requester. You will have to establish two simultaneous links: one for the CICS-to-CICS ISC, and one for the DB2-to-DB2 DRDA.

```
+----------------------+                  +---------------+
|        MVS/ESA       |                  |      AIX      |
+----------------------+                  +---------------+
|                      |                  |               |
|                      ===========/        |  CICS/6000    |
|    CICS/ESA  TOR     |         /        |               |
|                      |        /============    AOR      |
|                      |                  |               |
+----------------------+                  +-------||------+
|                      |                  |               |
|                      ===========/        |   DB2/6000    |
|    DB2 Remote DB     |         /        |               |
|       Server         |        /============    DDF      |
|                      |                  |               |
+----------------------+                  +---------------+
```

Figure 4. Remote CICS AOR with DRDA Database

4.1.4 Total AIX Environment (Scenario 2)

The final goal for downsizing is to migrate the network to TCP/IP, leaving a pure RISC System/6000 environment (see Figure 5). This is the schema used in our Scenario 2. The bigger effort will focus on the setup, definition, and customization of the TCP/IP terminal network and the retraining of your people for the new environment.

```
+---------------------------------------+
|      RISC SYSTEM/6000  AIX            |
+---------------+------------+----------+
|               |            |          |
|  TCP/IP       |            |          |
|   DCE         = CICS/6000  = DB2/6000 |
|  Encina       |            |          |
|               |            |          |
+---------------+------------+----------+
```

Figure 5. Scenario 2: Total AIX Environment

4.2 Our Network Configuration

Figure 6 on page 14 shows the network configuration we used to test scenarios 1 and 2. The RISC System/6000s are connected to a token-ring named USIBMSC, to which a VM/ESA* system, SCG20, is also connected. The VM system is used as a gateway to access the MVS system, WTSCMXA, through cross-domain definitions. The RISC System/6000s, BAFFIN, YELLOW, and AEGEAN, are all nodes in a DCE cell called gds. BAFFIN provides the DCE security and cell directory services for the cell. YELLOW has AIX SNA Server/6000 V2 installed and is acting as the PPC Gateway to AEGEAN, enabling AEGEAN to communicate with the MVS system, WTSCMXA, through LU 6.2 sessions. The

CICS region, SJA2109I, with all its components is run on AEGEAN. The DB2/6000 database product is also installed on AEGEAN.

```
+---------------+                     +-----------+----------+
|          1.1.2|                     | CICS/6000 | DB2/6000 |
| PPC Gateway   |                     | 1.1       | 1.1.0    |
| PPC EXEC      |                     |           |          |
|               |                     | SFS 1.1.1 |  COBOL   |
| SNA 2.1       |                     | LOG 1.1.1 |          |
|               |                     | PPC EXEC  |          |
+---------------+                     +-----------+----------+
|               |                     |                      |
| DCE Base 1.2  |                     |   DCE Base 1.2       |
|               |       USIBMSC       |                      |
| AIX 3.2.5     |                     |   AIX 3.2.5          |
+---------------*=====================*----------------------+
          YELLOW|*===================*|AEGEAN
                ||    TOKEN-RING     ||
          BAFFIN|*===================*|SCG20                        WTSCMXA
+---------------*=====================*-----------------------+       +---------------+
|               |                     |                       |       |               |
| DCE Security  |                     |        VM/ESA        =====/   |    MVS/ESA    |
| Server 1.2    |                     |        VTAM/VM        |   /=====   VTAM       |
|               |                     |                       |       +---------------+
| DCE CDS 1.2   |                     +-----------------------+       |               |
| DCE Base 1.2  |                                                     |  CICS 3.3.0   |
|               |                                                     |               |
| AIX 3.2.5     |                                                     |  DB2  3.1.0   |
+---------------+                                                     |               |
                                                                      +---------------+
```

Figure 6. Network Configuration

We recognize that the configuration we used to test the scenarios is not ideal, performance-wise. For example, the VM gateway is not required, as MVS can be connected directly to the token-ring network. Also, we could have provided a better distribution of the various components, for example, by placing the server component of DB2 in a separate machine. However, as we mentioned, the emphasis of our project was on application downsizing.

Chapter 5. Transferring Sequential Files

In this chapter we outline the different methods of transferring sequential files from the mainframe to an AIX workstation. We also discuss the various application components that need to be transferred. We explain the differences between the MVS and AIX file systems and show you how to set up an AIX file system and the appropriate directories for your project.

5.1 Preparing the File Transfer

Before you start transferring any files to the AIX system, you should complete the following checklist:

- Know which components must be transferred
  - Program source code
  - Copybooks
  - Maps
  - SQL create statements
  - CICS system definition (CSD) listing
  - Text or binary data files
- Create file systems and directories on your AIX machine
- Create a list of all host file names
- Create a list of the corresponding AIX file names
- Choose a file transfer means.

5.1.1 MVS Partitioned Data Set Members

Frequently the different source code programs that an MVS application uses are stored in a partitioned data set. There is no concept of a partitioned data set in AIX; AIX uses a hierarchical file system. You should create a directory for each partitioned data set and store all of the required members in that directory. This is the procedure you will use to transfer the program source code, the copybooks, and the BMS source for the maps.

If the volume of your data is large, we recommend that you create a new file system to hold the directories by using the command shown in Figure 7.

```
smit crjfs
```

Figure 7. Creating a File System

After creating the file system you must mount it in order to access it. Enter the command show in Figure 8.

```
mount /your/name
```

Figure 8. Mounting a File System

Now that your file system is accessible (that is, mounted) you can create the directories by entering the command shown in Figure 9 on page 16.

```
mkdir /your/name/newdirectory
```

Figure 9. Creating an AIX Directory

To generate a list of the data set members to transfer, you can use ISPF menu 3.1.

Option x generates a list of all member names in your ISPF list file. You have to leave ISPF and enter the KEEP option for the ISPF list data set before you can work with it. Afterwards you can use either the ISPF/PDF editor on MVS to generate the listing or any AIX tools such as sed, cut, or shell scripts.

5.1.2 AIX File Names

All of your programs, copybooks, and maps will have the same names. Make sure that these names are in the appropriate case, as AIX is case sensitive. Normally this will be uppercase, except for copybooks included with the SQL EXEC INCLUDE statement, which have to be in lowercase. You will need to have special extensions for some of these files, however, as shown in Table 1.

Table 1. Extensions for File Names

Type	**Extension**
COBOL program	.ccp
COBOL copybook	.cbl
BMS maps	.bms

5.1.3 SQL Create Statements

If you decide to transfer your DB2 data using BRIDGE/FASTLOAD For DB2/6000** or the export and import facilities through a DDCS connection, you will not have to transfer any create tablespace or create table statements. The different variants are discussed in Chapter 6, "Converting and Transferring DB2 Databases" on page 21. If you choose a different method, you may have to transfer the data and the create statements separately. In any case you will have to transfer all create statements for indexes and views separately.

It is important to have a list of all files containing the SQL statements that have to be transferred and to transfer those files in the same way as the program source code. If you no longer have the create statements, you can use various tools, such as the BACHMAN/DBA Catalog Extract for DB2**.

5.1.4 CICS Resource Definitions

If your application consists of only a few programs and maps, you might choose to enter these by hand. You can create a listing of the CICS/ESA* definitions by running the CICS DFHCSDUP utility on MVS, as shown in Figure 10 on page 17. This file can be transferred to the AIX system in the same way as the other sequential files. You can then either use it as a reference or write a program or script to generate the cicsadd statements for your CICS/6000 system. In A.5, "Hursley SupportPacs" on page 99 you can find a reference to a SupportPac from the IBM UK Laboratories Ltd, based at Hursley Park in England, offering the migration of the CICS definitions as a service.

```
//DFHCSDUP JOB (999,POK),'DFHCSDUP LISTINGS',
//             NOTIFY=&SYSUID,
//             CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1)
//*
//LISTCSD EXEC PGM=DFHCSDUP,REGION=1M
//*
//* LIST THE CSD CONTENTS
//*
//STEPLIB  DD DSN=CICS330.SDFHLOAD,DISP=SHR
//DFHCSD   DD DSN=CICS330.DFHCSD,DISP=SHR
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  LIST GROUP(yourgrp1) OBJECTS
  LIST GROUP(yourgrp2) OBJECTS
    ......
  LIST GROUP(DFHSTAND) OBJECTS
/*
//*
//
```

Figure 10. Sample DFHCSDUP JCL. The DFHSTAND group has all the PROFILEs addressed by the transaction and program definitions. All the desired definition groups must be extracted in the same run.

5.2 File Transfer Methods

Basically you have the following options for transferring files:

- TCP/IP File Transfer Protocol (FTP)
- IND$FILE
- Diskette.

You can also combine these methods; for example, you could use the SNA IND$FILE to transfer a file from MVS to OS/2* and use the TCP/IP FTP program to copy the files to your AIX system.

5.2.1 TCP/IP File Transfer Protocol

Probably the most efficient file transfer method is the TCP/IP FTP. All AIX systems come with complete TCP/IP support. To use FTP you need to have TCP/IP installed on the MVS mainframe. This is a separate program product. The data transfer rate you achieve with FTP depends on your connection to the mainframe system; however, the rate will be significantly faster than with the SNA IND$FILE file transfer method.

Figure 11 on page 18 shows you how to use the TCP/IP FTP to transfer your sequential files.

```
#ftp 9.12.15.5
Connected to 9.12.15.5.
220-FTPSERVE IBM MVS V2R2.1 at WTSCMXA.ITSC.POK.IBM.COM, 16:26:39 on 06/14/94
220 Connection will close if idle for more than 5 minutes.
Name (9.12.15.5:root): carlo3
331 Send password please.
Password:
230 CARLO3 is logged on.
ftp> ascii
200 Representation type is ASCII.
ftp> get 'host.file.name(member)' /aix/directory/filename.extension
200 Port request OK.
125 Sending data set HOST.FILE.NAME(MEMBER) FIXrecfm 80
250 Transfer completed successfully.
431 bytes received in 0.116 seconds (3.628 Kbytes/s)
ftp> quit
221 Quit command received. Goodbye.
#
```

Figure 11. Sample FTP Session

The userid and password that you specify must be known to the MVS system. The userid must be authorized to read the file you want to transfer. If the file you want to transfer is text, you have to set the type to ascii to provide automatic conversion from EBCDIC to ASCII.

For large applications it can be tedious to transfer each file manually, so FTP provides you with macros to simplify the bulk transfer of many files. Figure 12 shows you a sample .netrc file, which must be in the home directory (normally /home/root/.netrc) of the user invoking FTP. The macro used in this file is the init macro, which is automatically processed once the connection is established.

```
machine 9.12.15.5 login carlo3 password yourpw
macdef init
binary
get HOST.FILE1      /aix/directory/FILE1.extension
get HOST.FILE2      /aix/directory/FILE2.extension
get HOST.FILE3      /aix/directory/FILE3.extension
get HOST.FILE4      /aix/directory/FILE4.extension
get HOST.FILEN      /aix/directory/FILEN.extension
bye
```

Figure 12. Sample .netrc File. Please note that the password is not encrypted and can be read by anybody who has read access to this file.

Please refer to the FTP documentation for a detailed list of all features that FTP offers you.

Note

Even though the IP address of the machine is in our .netrc file, we still have to enter this address when invoking FTP.

5.2.2 SNA IND$FILE

Even if TCP/IP is not available on the mainframe system you can still use a direct file transfer to the AIX system. You need to install an SNA link to your host system and define at least one LU2 session. You can use the Host Connection Program/6000 (hcon) as your 3270 emulation program. The small shell script in Figure 13 is provided as an example of bulk file transfer using SNA IND$FILE.

```
#!/bin/ksh
fxfer -n a -t -H TSO -d "'HOST.FILE.NAME(MEMBER)'" /aix/directory/FILE.extension
sleep 3
fxfer -n a -t -H TSO -d "'HOST.FILE.NAM1(MEMBER)'" /aix/directory/FIL1.extension
sleep 3
fxfer -n a -t -H TSO -d "'HOST.FILE.NAM2(MEMBER)'" /aix/directory/FIL2.extension
```

Figure 13. Sample fxfer Script

The fxfer script will run only if you have previously logged in to a TSO system in your hcon session a. To run this script you must enter option 6 on the ISPF primary menu or return to the TSO READY prompt by exiting ISPF. The -t flag tells fxfer to convert data from EBCDIC to ASCII. For transferring binary files you will want to remove this flag. For a complete description of all possible flags refer to the fxfer documentation.

During the file transfer you will not be able to use the hcon session for other purposes. For this reason you may want to set up more than one LU2 session and start these from different aixterms. You can start an hcon session from any aixterm by entering the command as shown in Figure 14.

```
e789 a
```

Figure 14. Starting a 3270 Terminal Emulation Session

The argument *a* specifies the name of the session you want to start from this aixterm.

5.2.3 Diskette

If your AIX system is not attached to any network or you have chosen not to implement the SNA link and do not have any TCP/IP connection, your only option is to transfer the data using diskettes. Normally you will have a programmable workstation (PWS) connected to your MVS mainframe. You can then download your files to your PWS and copy them onto 1.44MB or 2.88MB diskettes, depending on the diskette drives of both your PWS and your IBM RISC System/6000. Diskettes written in the DOS format can be read on the AIX system by using the dosread command.

You will not be able to transfer files that are too large to fit on a single diskette in this way, as AIX and the PWS operating systems use different archiving programs. It is also the slowest way of transferring data, because two steps are involved in the process.

You can read all of the data on a diskette using one command—xargs—provided that all of the files are either binary or text (see Figure 15 on page 20).

```
#dosdir | xargs -t -i dosread -a {} {}
```

Figure 15. Reading All Files from a Diskette

The output of the dosdir command is piped into the xargs command, which executes the dosread command for every input line. The -a flag tells the dosread program to convert the carriage return and line feed of the file to just line feed. You will want to remove the -a flag if you are reading binary files.

5.2.4 Tape

You can attach 3480 tape devices to an IBM RISC System/6000. Using an attached 3480 tape device you can read tape cartridges written on an MVS system. The 3480 tape devices are more expensive than the other available tape devices for the IBM RISC System/6000. The 3480 tape device provides a good means of transferring files from your MVS system to your AIX system, but we recommend using one of the previously described means of transferring files if you do not already have a 3480 tape device.

Chapter 6. Converting and Transferring DB2 Databases

In this chapter we examine the various ways of converting and transferring DB2 data from the MVS mainframe to the AIX system. We look at:

- DB2/6000′s DDCS/6000
- BRIDGE/FASTLOAD
- Conventional unload and reload facilities of DB2 MVS and DB2/6000.

With the exception of the DDCS/6000 solution, you have to transfer the data and/or your table definitions manually. Chapter 5, "Transferring Sequential Files" on page 15 explains how to do that. To transfer the SQL statements you have to set the transfer mode to text, and for the data itself you have to use binary.

If you use referential integrity, we suggest that you transfer your SQL create table statements separately and create the tables in your DB2/6000 database before transferring the data. You should take this appraoch even if you are using DDCS/6000 or BRIDGE/FASTLOAD, because neither tool supports the transfer of DB2 catalog information for referential integrity. Another reason to use this approach is that both tools change all column definitions with *NOT NULL WITH DEFAULT* to *NOT NULL*.

6.1 DDCS/6000

DDCS is the IBM implementation of DRDA, and DDCS/6000 is the IBM RISC System/6000 version of DDCS. DDCS/6000 allows you to access databases on remote systems. To be able to use DDCS/6000 to transfer your DB2 data you must have set up the SNA link and an LU6.2 connection to your MVS system, through VTAM. On the MVS side you have to define and conveniently populate the communications database of the DB2 subsystem on which your data resides. Your LU6.2 connection must correspond to the LU6.2 definition of the MVS DB2 subsystem. Please refer to the DDCS publications listed in the preface for more detailed information about how to establish the DDCS link.

Before you can access the remote database you must catalog it (see Figure 16).

```
#!/bin/ksh
db2 CATALOG CPIC NODE yournode REMOTE yourcpic SECURITY PROGRAM
db2 CATALOG DATABASE yourdb AS yourdb AT NODE yournode AUTHENTICATION DCS
db2 CATALOG DCS DATABASE yourdb AS hostdbloc
db2 TERMINATE
export DDCSSETP=″-f=NUL -s=e″
db2 CONNECT TO yourdb USER mvsdb2userid USING password
db2 BIND /home/yourinstance/sqllib/bnd/@ddcsbind.lst BLOCKING ALL
db2 CONNECT RESET
unset DDCSSETP
db2 TERMINATE
```

Figure 16. Sample Catalog of Remote Database

To execute the catalog shell script you must be logged in as the DB2 instance owner. You also need access to an MVS userid that is authorized to issue the bind command and access the application tables.

Once the remote database is cataloged and thus known in your local system you can use the DB2/6000 export and import commands to transfer the DB2 data. Depending on the format you choose, the table definitions will be included or not included in the generated files. Figure 17 on page 22 provides you with an example of a shell script to run the export and import commands for the integrated exchange format (IXF), which contains the data and the table definitions.

```
#!/bin/ksh
db2 CONNECT TO yourremotedb USER mvsdb2userid USING password
db2 EXPORT TO /aix/directory/file OF IXF SELECT '*' FROM yourtable
db2 EXPORT TO /aix/directory/file2 OF IXF SELECT '*' FROM yourtable2
db2 CONNECT RESET
db2 CONNECT TO yourlocaldb
db2 IMPORT FROM /aix/directory/file OF IXF CREATE INTO yourtable
db2 IMPORT FROM /aix/directory/file2 OF IXF CREATE INTO yourtable2
db2 CONNECT RESET
```

Figure 17. Sample DDCS Export and Import Script

To run the export and import script you must be logged in to AIX with a userid authorized to create the tables with the required qualifier in your local DB2/6000 system. If you are exporting numerous large tables, you can create a file system to hold the unload files. Once you have imported the exported files you can remove the whole file system again and thus easily reclaim the disk space. Section 5.1.1, “MVS Partitioned Data Set Members” on page 15 explains how to create a file system.

If you choose to create your tables before importing the data, you must replace the CREATE parameter with INSERT on the IMPORT command.

Using DDCS/6000 is probably the most efficient way for you to transfer your DB2 data. DDCS/6000 does all of the conversion work from EBCDIC to ASCII. The only drawback is that you need to set up an SNA link to your mainframe system, which, for the reasons described in Chapter 3, “Recommendations” on page 9, we recommend that you do anyway.

Note

In the current versions of DB2/6000 and DDCS/6000 all table definitions for columns defined as *NOT NULL WITH DEFAULT* are changed to *NOT NULL*. In certain cases this can cause compilation errors or run-time errors for your application programs. Unfortunately these column definitions cannot be changed through an *ALTER* statement, so you have to:

1. Export your table data.
2. Drop your existing DB2/6000 table.
3. Change and run your create table statement.
4. Re-create all dependent views, indexes, and grants.
5. Import the table data.
6. Rebind all dependent plans.

6.2 BRIDGE/FASTLOAD

BRIDGE/FASTLOAD from Bridge Technology Inc. enables you to unload the table data on the mainframe side and reload it incredibly quickly on the workstation side. In this project we used BRIDGE/FASTLOAD to transfer our DB2 data.

On the MVS side you have to install the BRIDGE/FASTLOAD for DB2/6000 Mainframe Edition to create the unload files. These files are created in a DB2/6000 internal format and need not be converted.

The BRIDGE/FASTLOAD for DB2/6000 Server Edition needs to be installed on your workstation. This product loads the unload files created on the mainframe into your local DB2/6000 database. As all of the data already is in an internal DB2/6000 format, this process takes very little time.

The installation procedure for both products is documented in the BRIDGE/FASTLOAD manuals. We do not cover the installation in this book. Please note that the mainframe product comes on an AIX diskette and is in the tape archive (tar) format. You have to unpack the product on an IBM RISC System/6000 and then transfer it to your mainframe.

After completing the customization of the procedure for running BRIDGE/FASTLOAD, you can unload all of your tables. Figure 18 shows a simple REXX procedure to run the program for multiple tables.

```
/*  REXX                                                        */
/*--------------------------------------------------------------*/
/* set up an array of all your tables that you want to unload   */
/*--------------------------------------------------------------*/
SW.1 = ″SELECT * FROM YOUR.TABLE1″
SW.2 = ″SELECT * FROM YOUR.TABLE2″
SW.3 = ″SELECT * FROM YOUR.TABLE3″
SW.4 = ″SELECT * FROM YOUR.TABLE4″
SW.5 = ″SELECT * FROM YOUR.TABLE5″
SW.6 = ″SELECT * FROM YOUR.TABLE6″
SW.7 = ″SELECT * FROM YOUR.TABLE7″
SW.0 = 7

/*--------------------------------------------------------------*/
/* loop through all tables and invoke BRIDGE/FASTLOAD           */
/*--------------------------------------------------------------*/
do c=1 to SW.0
  ″EXEC ′YOURLIB.FASTLOAD.CMD(FLEX20)′ ′/SS″″″ || SW.c || ″″″′ EXEC″
  table = substr(SW.c,lastpos(′.′,SW.c)+1)

/*--------------------------------------------------------------*/
/* rename the created files so that they do not get overwritten */
/*--------------------------------------------------------------*/
  ″rename fastload.d00″ table || ″.D00″
  ″rename fastload.pkg″ table || ″.PKG″
  ″rename fastload.l00″ table || ″.L00″
end
exit
```

Figure 18. Simple REXX Procedure for Multiple Invocations of BRIDGE/FASTLOAD

To run the script you can create a batch job similar to the job shown in Figure 19 on page 24.

```
//USERID#    JOB (999,POK),'DB2 V3',NOTIFY=&SYSUID,
//         CLASS=A,MSGCLASS=T,REGION=5000K,
//         MSGLEVEL=(1,1)
//JOBLIB   DD  DSN=DB2.DSNLOAD,DISP=SHR
//         DD  DSN=YOURLIB.FASTLOAD.LOAD,DISP=SHR
//PH01PS02 EXEC PGM=IKJEFT01,DYNAMNBR=100
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSTSIN  DD  *
EXEC 'YOURLIB.FASTLOAD.CMD(RUNALL)' EXEC
```

Figure 19. Sample Batch Job for Running BRIDGE/FASTLOAD

If you run the REXX procedure as shown you will get three files for each table you unload. Table 2 contains a description of these files. The first level of each file is your userid, and the second level is the table name (without the qualifier).

Table 2. BRIDGE/FASTLOAD Unload Files

Third Level	**Description**
D00	Extracted data
L00	Extracted data for long varchar fields
PKG	Descriptive file for load program

BRIDGE/FASTLOAD provides no means for the actual transfer of the unload files to your workstation. You can transfer all of the unload files in binary mode using the procedures documented in Chapter 5, "Transferring Sequential Files" on page 15.

Once you have received all of the files on your workstation you can run the BRIDGE/FASTLOAD load program. For each table separately or for all tables together you can run the script shown in Figure 20.

```
#!/bin/ksh
db2 CONNECT TO yourlocaldb
export PATH=$PATH:/bridge/directory
cd /your/download/directory
cloadex /Tyourlocaldb:yourqualifier.yourtable /R-1 /Iyourtable >yourtable.out
cloadex /Tyourlocaldb:yourqualifier.yourtable2 /R-1 /Iyourtable2 >yourtable2.out
db2 CONNECT RESET
```

Figure 20. cloadex Script

The cloadex script creates the DB2 tables and loads all rows of the data into them. Run this script with a userid that is authorized to create the DB2 tables with your qualifier in your database. Remember to check the output of each table in the yourtable.out file for errors or warnings.

If your DB2 tables already exist when you run the load program, existing table definitions are used, provided that they correspond to the data you intend to load.

Note

In the current version of BRIDGE/FASTLOAD all table definitions for columns defined as *NOT NULL WITH DEFAULT* are changed to *NOT NULL*. In certain cases this can cause compilation errors or run-time errors for your application programs. Unfortunately these column definitions cannot be changed through an *ALTER* statement. 6.1, "DDCS/6000" on page 21 shows you how to proceed if this happens in your downsizing project.

6.3 Unload Programs

If you cannot use DDCS/6000 or BRIDGE/FASTLOAD, your only other option short of writing your own complete mechanism is to use IBM′s Query Management Facility (QMF)* or a DB2 unload program such as DSNTIAUL or DSNTEP2 to unload the data to a sequential file on your MVS system. To produce a valid load format for DB2/6000 you have to edit or write a program to convert the files generated by either unload program.

We do not recommend transferring your DB2 data using an unload program. The effort will be much larger than if you use DDCS/6000 or BRIDGE/FASTLOAD. In the section that follows, however, we show you how to completely transfer and load data extracted with the QMF export command, and we briefly review other available unload programs.

6.3.1 QMF Export

If you have QMF on your MVS system, you can use its export command to unload your DB2 data to a sequential file, as follows:

1. Invoke QMF.
2. Run the QMF command *DI YOUR.TABLE*.
3. Run the QMF command *EXPORT DATA TO YOUR.DATASET* with the options *DATAFORMAT=IXF* and *OUTPUTMODE=CHARACTER* .

You can enter the options for the QMF export command using the EXPORT command prompt as shown in Figure 21 on page 26.

```
+----------------------------------------------------------------------------+
|                              EXPORT command prompt                         |
|                                                              1  to 16 of 16 |
| EXPORT type DATA                                                           |
|        name                                                                |
|                                                                            |
| TO                                                                         |
| Data set name ( YOUR.DATASET.NAME                                   )      |
|               Enter the name of the data set into which the               |
|               object is to be copied.                                      |
| Member        (          ) If the data set is partitioned,                 |
|                            enter the member name.                          |
| Confirm       ( YES      ) Display the confirmation panel before replacing |
|                            an existing data set?  YES or NO.               |
| Dataformat    ( IXF      ) The data format to use for your exported        |
|                            data:  QMF or IXF.                              |
| Outputmode    ( CHARACTER  ) Valid only when DATAFORMAT is IXF. The        |
|                              output mode used for numeric data in your     |
|                              exported data:  BINARY or CHARACTER.          |
+----------------------------------------------------------------------------+
| F1=Help  F3=End  F7=Backward  F8=Forward                                   |
+----------------------------------------------------------------------------+
```

Figure 21. QMF EXPORT Command Prompt

You can now transfer the YOUR.DATASET.NAME dataset to your workstation using the text mode, as described in Chapter 5, “Transferring Sequential Files” on page 15. The unload file needs to be edited because the first lines contain a description of the table itself with the columns. You can edit on either the MVS or AIX system, wherever you feel more comfortable.

To load the data you must first create the DB2/6000 table in your database. You must then make a list of the starting and ending position of each column in your load file. When you have done this you can run the DB2/6000 import command to load the data (see Figure 22).

```
#!/bin/ksh
db2 CONNECT TO yourdb
db2 ″IMPORT FROM your/data/file OF ASC MODIFIED BY T METHOD L \
(1 5,7 16,18 22,24 28, 31 35, 38 46,48 55) MESSAGES msgs.txt INSERT \
INTO your.table″
db2 COMMIT
db2 CONNECT RESET
```

Figure 22. Script to Import ASCII Data into DB2/6000

The L option of the METHOD parameter means that the subsequent specifications contain the starting and ending position for each column. Run this script from a userid authorized to load the DB2/6000 table in your database. After running the script check the msgs.txt file for warnings, errors, and information messages.

6.3.2 DSNTIAUL

DSNTIAUL is an assembler language sample program provided in the DB2 installation verification procedure (IVP). DSNTIAUL unloads up to 100 tables into sequential data sets and produces the load statements to reload those tables. Figure 23 shows an example of the JCL you need to run DSNTIAUL. Figure 24, Figure 25 on page 28, and Figure 26 on page 29 show the sample output listings. You can find a complete explanation of this program in the *DB2 V3 Administration Guide*.

With DSNTIAUL you can extract data and definitions from the host DB2, transfer the sequential files to the AIX system, and re-create the tables in the DB2/6000 environment.

```
//jobname JOB (999,POK),CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1)
//JOBLIB  DD  DSN=DSN310.SDSNLOAD,DISP=SHR
//*
//PH01S01 EXEC PGM=IKJEFT01,DYNAMNBR=20
//SYSTSPRT DD SYSOUT=*
//SYSTSIN  DD *
 DSN SYSTEM(DB3A)
 RUN  PROGRAM(DSNTIAUL) PLAN(DSNTIB31) PARMS('SQL') -
      LIB('DSN310.RUNLIB.LOAD')
/*
//SYSPRINT DD SYSOUT=*
//SYSUDUMP DD SYSOUT=*
//SYSPUNCH DD DSN=syspunch.dataset,DISP=(,CATLG,DELETE),
// VOL=SER=xxxxxx,UNIT=SYSDA,SPACE=(TRK,(1,1))
//SYSIN    DD *
  SELECT * FROM DSN8310.DEPT;
  SELECT * FROM DSN8310.EMP;
  SELECT * FROM DSN8310.PROJ;
/*
//*
//SYSREC00 DD DSN=sysrec00.dataset,DISP=(,CATLG,DELETE),
// VOL=SER=xxxxxx,UNIT=SYSDA,SPACE=(TRK,(1,1))
//SYSREC01 DD DSN=sysrec01.dataset,DISP=(,CATLG,DELETE),
// VOL=SER=xxxxxx,UNIT=SYSDA,SPACE=(TRK,(1,1))
//SYSREC02 DD DSN=sysrec02.dataset,DISP=(,CATLG,DELETE),
// VOL=SER=xxxxxx,UNIT=SYSDA,SPACE=(TRK,(1,1))
```

Figure 23. DSNTIAUL JCL. There is one SYSRECnn DD statement per table SELECTed, and no DCB is allowed.

```
READY
 DSN SYSTEM(DB3A)
DSN
 RUN PROGRAM(DSNTIAUL) PLAN(DSNTIB31) PARMS('SQL') LIB('DSN310.RUNLIB.LOAD')
DSN
END
```

Figure 24. DSNTIAUL SYSTSPRT Output

```
DSNT505I DSNTIAUL OPTIONS USED: SQL
DSNT490I SAMPLE DATA UNLOAD PROGRAM
DSNT503I UNLOAD DATA SET SYSREC00 RECORD LENGTH SET TO    68
DSNT504I UNLOAD DATA SET SYSREC00 BLOCK SIZE SET TO 23460
DSNT495I SUCCESSFUL UNLOAD           14 ROWS OF TABLE DSN8310.DEPT
DSNT503I UNLOAD DATA SET SYSREC01 RECORD LENGTH SET TO   101
DSNT504I UNLOAD DATA SET SYSREC01 BLOCK SIZE SET TO 23432
DSNT495I SUCCESSFUL UNLOAD           42 ROWS OF TABLE DSN8310.EMP
DSNT503I UNLOAD DATA SET SYSREC02 RECORD LENGTH SET TO    74
DSNT504I UNLOAD DATA SET SYSREC02 BLOCK SIZE SET TO 23458
DSNT495I SUCCESSFUL UNLOAD           20 ROWS OF TABLE DSN8310.PROJ
```

Figure 25. DSNTIAUL SYSPRINT Output

```
LOAD DATA LOG NO INDDN SYSREC00 INTO TABLE
    DSN8310.DEPT
 (
 DEPTNO                                    POSITION(      1        )
 CHAR(                     3) ,
 DEPTNAME                                  POSITION(      4        )
 VARCHAR                      ,
 MGRNO                                     POSITION(     42        )
 CHAR(                     6)
      NULLIF(      48)='?',
 ADMRDEPT                                  POSITION(     49        )
 CHAR(                     3) ,
 LOCATION                                  POSITION(     52        )
 CHAR(                    16)
      NULLIF(      68)='?'
 )
LOAD DATA LOG NO INDDN SYSREC01 INTO TABLE
    DSN8310.EMP
 (
 EMPNO                                     POSITION(      1        )
 CHAR(                     6) ,
 FIRSTNME                                  POSITION(      7        )
 VARCHAR                      ,
 MIDINIT                                   POSITION(     21        )
 CHAR(                     1) ,
 LASTNAME                                  POSITION(     22        )
 VARCHAR                      ,
 WORKDEPT                                  POSITION(     39        )
 CHAR(                     3)
      NULLIF(      42)='?',
 PHONENO                                   POSITION(     43        )
 CHAR(                     4)
      NULLIF(      47)='?',
 HIREDATE                                  POSITION(     48        )
 DATE EXTERNAL(           10)
      NULLIF(      58)='?',
 JOB                                       POSITION(     59        )
 CHAR(                     8)
      NULLIF(      67)='?',
 EDLEVEL                                   POSITION(     68        )
 SMALLINT
      NULLIF(      70)='?',
 SEX                                       POSITION(     71        )
 CHAR(                     1)
      NULLIF(      72)='?',
 BIRTHDATE                                 POSITION(     73        )
 DATE EXTERNAL(           10)
      NULLIF(      83)='?',
 SALARY                                    POSITION(     84:     88)
 DECIMAL
      NULLIF(      89)='?',
 BONUS                                     POSITION(     90:     94)
 DECIMAL
      NULLIF(      95)='?',
 COMM                                      POSITION(     96:    100)
 DECIMAL
      NULLIF(     101)='?'
 )
```

Figure 26 (Part 1 of 2). DSNTIAUL SYSPUNCH Output

```
LOAD DATA LOG NO INDDN SYSREC02 INTO TABLE
    DSN8310.PROJ
 (
 PROJNO                                 POSITION(       1         )
 CHAR(                      6) ,
 PROJNAME                               POSITION(       7         )
 VARCHAR                       ,
 DEPTNO                                 POSITION(      33         )
 CHAR(                      3) ,
 RESPEMP                                POSITION(      36         )
 CHAR(                      6) ,
 PRSTAFF                                POSITION(      42:      44)
 DECIMAL
      NULLIF(      45)='?',
 PRSTDATE                               POSITION(      46         )
 DATE EXTERNAL(            10)
      NULLIF(      56)='?',
 PRENDATE                               POSITION(      57         )
 DATE EXTERNAL(            10)
      NULLIF(      67)='?',
 MAJPROJ                                POSITION(      68         )
 CHAR(                      6)
      NULLIF(      74)='?'
 )
```

Figure 26 (Part 2 of 2). DSNTIAUL SYSPUNCH Output

If your tables contain binary or decimal fields, you must use the SQL functions CHAR and DECIMAL to convert the fields to characters.

6.3.3 DSNTEP2

Another way of creating sequential files from your DB2 MVS data is to use the sample PLI program, DSNTEP2, which is provided in the DB2 IVP. This program handles both data definition language (DDL) and data manipulation language (DML) statements (including SELECT). You can enter any SQL statement that will be used to produce the unload file. You also have to edit the file manually or by running a program that deals with the output listing, before it can be used to load into DB2/6000.

Figure 27 on page 31 shows an example of the JCL you need to run DSNTEP2. Figure 28 on page 31 and Figure 29 on page 32 show the sample output listings. You can find a complete explanation of DSNTEP2 in the *DB2 V3 Administration Guide*.

```
//DSNTEP2 JOB (999,POK),NOTIFY=&SYSUID,
//        CLASS=A,MSGCLASS=T,REGION=5000K,
//        MSGLEVEL=(1,1)
//****************************************************
//JOBLIB   DD  DSN=DSN310.SDSNLOAD,DISP=SHR
//         DD  DSN=PLI.V2R3M0.PLICOMP,DISP=SHR
//         DD  DSN=PLI.V2R3M0.PLILINK,DISP=SHR
//         DD  DSN=PLI.V2R3M0.SIBMLINK,DISP=SHR
//         DD  DSN=PLI.V2R3M0.PLIBASE,DISP=SHR
//         DD  DSN=PLI.V2R3M0.SIBMBASE,DISP=SHR
//PH01PS02 EXEC PGM=IKJEFT01,DYNAMNBR=20
//DBRMLIB  DD  DSN=DSN310.DBRMLIB.DATA,DISP=SHR
//SYSTSPRT DD  SYSOUT=*
//SYSPRINT DD  SYSOUT=*
//SYSUDUMP DD  SYSOUT=*
//SYSTSIN  DD  *
 DSN SYSTEM(DB3A)
 RUN  PROGRAM(DSNTEP2) PLAN(DSNTEP31)-
      LIB('DSN310.RUNLIB.LOAD')
 END
/*
//SYSIN    DD *
    SELECT * FROM DSN8310.EMP;
/*
//*
```

Figure 27. DSNTEP2 JCL

```
READY
 DSN SYSTEM(DB3A)
DSN
 RUN PROGRAM(DSNTEP2) PLAN(DSNTEP31) LIB('DSN310.RUNLIB.LOAD')
DSN
 END
READY
END
```

Figure 28. DSNTEP2 SYSTSPRT Output

```
PAGE    1
**INPUT STATEMENT:  SELECT * FROM DSN8310.EMP;



          +------------------------------------------------------------------------------------------------------------------
          | EMPNO  |   FIRSTNME   | MIDINIT |    LASTNAME      | WORKDEPT | PHONENO |  HIREDATE  |   JOB    |  EDLEVEL  | SEX |
          +------------------------------------------------------------------------------------------------------------------
        1_| 000010 | CHRISTINE    | I       | HAAS             | A00      | 3978    | 01/01/1965 | PRES     |        18 | F   |
        2_| 000020 | MICHAEL      | L       | THOMPSON         | B01      | 3476    | 10/10/1973 | MANAGER  |        18 | M   |
        3_| 000030 | SALLY        | A       | KWAN             | C01      | 4738    | 04/05/1975 | MANAGER  |        20 | F   |
        4_| 000050 | JOHN         | B       | GEYER            | E01      | 6789    | 08/17/1949 | MANAGER  |        16 | M   |
        5_| 000060 | IRVING       | F       | STERN            | D11      | 6423    | 09/14/1973 | MANAGER  |        16 | M   |
        6_| 000070 | EVA          | D       | PULASKI          | D21      | 7831    | 09/30/1980 | MANAGER  |        16 | F   |
        7_| 000090 | EILEEN       | W       | HENDERSON        | E11      | 5498    | 08/15/1970 | MANAGER  |        16 | F   |
        8_| 000100 | THEODORE     | Q       | SPENSER          | E21      | 0972    | 06/19/1980 | MANAGER  |        14 | M   |
        9_| 000110 | VINCENZO     | G       | LUCCHESI         | A00      | 3490    | 05/16/1958 | SALESREP |        19 | M   |
       10_| 000120 | SEAN         |         | O'CONNELL        | A00      | 2167    | 12/05/1963 | CLERK    |        14 | M   |
       11_| 000130 | DOLORES      | M       | QUINTANA         | C01      | 4578    | 07/28/1971 | ANALYST  |        16 | F   |
       12_| 000140 | HEATHER      | A       | NICHOLLS         | C01      | 1793    | 12/15/1976 | ANALYST  |        18 | F   |
       13_| 000150 | BRUCE        |         | ADAMSON          | D11      | 4510    | 02/12/1972 | DESIGNER |        16 | M   |
       14_| 000160 | ELIZABETH    | R       | PIANKA           | D11      | 3782    | 10/11/1977 | DESIGNER |        17 | F   |
       15_| 000170 | MASATOSHI    | J       | YOSHIMURA        | D11      | 2890    | 09/15/1978 | DESIGNER |        16 | M   |
       16_| 000180 | MARILYN      | S       | SCOUTTEN         | D11      | 1682    | 07/07/1973 | DESIGNER |        17 | F   |
       17_| 000190 | JAMES        | H       | WALKER           | D11      | 2986    | 07/26/1974 | DESIGNER |        16 | M   |
       18_| 000200 | DAVID        |         | BROWN            | D11      | 4501    | 03/03/1966 | DESIGNER |        16 | M   |
       19_| 000210 | WILLIAM      | T       | JONES            | D11      | 0942    | 04/11/1979 | DESIGNER |        17 | M   |
       20_| 000220 | JENNIFER     | K       | LUTZ             | D11      | 0672    | 08/29/1968 | DESIGNER |        18 | F   |
       21_| 000230 | JAMES        | J       | JEFFERSON        | D21      | 4265    | 11/21/1966 | CLGNER   | ?         | ?   |
       22_| 000240 | SALVATORE    | M       | MARINO           | D21      | 3780    | 12/05/1979 | CLERK    |        17 | M   |
       23_| 000250 | DANIEL       | S       | SMITH            | D21      | 0961    | 10/30/1969 | CLERK    |        15 | M   |
       24_| 000260 | SYBIL        | V       | JOHNSON          | D21      | 8953    | 09/11/1975 | CLERK    |        16 | F   |
       25_| 000270 | MARIA        | L       | PEREZ            | D21      | 9001    | 09/30/1980 | CLERK    |        15 | F   |
       26_| 000280 | ETHEL        | R       | SCHNEIDER        | E11      | 8997    | 03/24/1967 | OPERATOR |        17 | F   |
       27_| 000290 | JOHN         | R       | PARKER           | E11      | 4502    | 05/30/1980 | OPERATOR |        12 | M   |
       28_| 000300 | PHILIP       | X       | SMITH            | E11      | 2095    | 06/19/1972 | OPERATOR |        14 | M   |
       29_| 000310 | MAUDE        | F       | SETRIGHT         | E11      | 3332    | 09/12/1964 | OPERATOR |        12 | F   |
       30_| 000320 | RAMLAL       | V       | MEHTA            | E21      | 9990    | 07/07/1965 | FIELDREP |        16 | M   |
       31_| 000330 | WING         |         | LEE              | E21      | 2103    | 02/23/1976 | FIELDREP |        14 | M   |
       32_| 000340 | JASON        | R       | GOUNOT           | E21      | 5698    | 05/05/1947 | FIELDREP |        16 | M   |
       33_| 200010 | DIAN         | J       | HEMMINGER        | A00      | 3978    | 01/01/1965 | SALESREP |        18 | F   |
       34_| 200120 | GREG         |         | ORLANDO          | A00      | 2167    | 05/05/1972 | CLERK    |        14 | M   |
       35_| 200140 | KIM          | N       | NATZ             | C01      | 1793    | 12/15/1976 | ANALYST  |        18 | F   |
       36_| 200170 | KIYOSHI      |         | YAMAMOTO         | D11      | 2890    | 09/15/1978 | DESIGNER |        16 | M   |
       37_| 200220 | REBA         | K       | JOHN             | D11      | 0672    | 08/29/1968 | DESIGNER |        18 | F   |
       38_| 200240 | ROBERT       | M       | MONTEVERDE       | D21      | 3780    | 12/05/1979 | CLERK    |        17 | M   |
       39_| 200280 | EILEEN       | R       | SCHWARTZ         | E11      | 8997    | 03/24/1967 | OPERATOR |        17 | F   |
       40_| 200310 | MICHELLE     | F       | SPRINGER         | E11      | 3332    | 09/12/1964 | OPERATOR |        12 | F   |
       41_| 200330 | HELENA       |         | WONG             | E21      | 2103    | 02/23/1976 | FIELDREP |        14 | F   |
       42_| 200340 | ROY          | R       | ALONZO           | E21      | 5698    | 05/05/1947 | FIELDREP |        16 | M   |
          +------------------------------------------------------------------------------------------------------------------
```

Figure 29 (Part 1 of 2). DSNTEP2 SYSPRINT Output

```
PAGE    2
                                     ---------------------------------------------------------+
                                     | BIRTHDATE |    SALARY     |    BONUS      |     COMM      |
                                     ---------------------------------------------------------+
                                  1_| 08/14/1933 |    52750.00   |     1000.00   |     4220.00   |
                                  2_| 02/02/1948 |    41250.00   |      800.00   |     3300.00   |
                                  3_| 05/11/1941 |    38250.00   |      800.00   |     3060.00   |
                                  4_| 09/15/1925 |    40175.00   |      800.00   |     3214.00   |
                                  5_| 07/07/1945 |    32250.00   |      600.00   |     2580.00   |
                                  6_| 05/26/1953 |    36170.00   |      700.00   |     2893.00   |
                                  7_| 05/15/1941 |    29750.00   |      600.00   |     2380.00   |
                                  8_| 12/18/1956 |    26150.00   |      500.00   |     2092.00   |
                                  9_| 11/05/1929 |    46500.00   |      900.00   |     3720.00   |
                                 10_| 10/18/1942 |    29250.00   |      600.00   |     2340.00   |
                                 11_| 09/15/1925 |    23800.00   |      500.00   |     1904.00   |
                                 12_| 01/19/1946 |    28420.00   |      600.00   |     2274.00   |
                                 13_| 05/17/1947 |    25280.00   |      500.00   |     2022.00   |
                                 14_| 04/12/1955 |    22250.00   |      400.00   |     1780.00   |
                                 15_| 01/05/1951 |    24680.00   |      500.00   |     1974.00   |
                                 16_| 02/21/1949 |    21340.00   |      500.00   |     1707.00   |
                                 17_| 06/25/1952 |    20450.00   |      400.00   |     1636.00   |
                                 18_| 05/29/1941 |    27740.00   |      600.00   |     2217.00   |
                                 19_| 02/23/1953 |    18270.00   |      400.00   |     1462.00   |
                                 20_| 03/19/1948 |    29840.00   |      600.00   |     2387.00   |
                                 21_| ?          | ?             | ?             | ?             |
                                 22_| 03/31/1954 |    28760.00   |      600.00   |     2301.00   |
                                 23_| 11/12/1939 |    19180.00   |      400.00   |     1534.00   |
                                 24_| 10/05/1936 |    17250.00   |      300.00   |     1380.00   |
                                 25_| 05/26/1953 |    27380.00   |      500.00   |     2190.00   |
                                 26_| 03/28/1936 |    26250.00   |      500.00   |     2100.00   |
                                 27_| 07/09/1946 |    15340.00   |      300.00   |     1227.00   |
                                 28_| 10/27/1936 |    17750.00   |      400.00   |     1420.00   |
                                 29_| 04/21/1931 |    15900.00   |      300.00   |     1272.00   |
                                 30_| 08/11/1932 |    19950.00   |      400.00   |     1596.00   |
                                 31_| 07/18/1941 |    25370.00   |      500.00   |     2030.00   |
                                 32_| 05/17/1926 |    23840.00   |      500.00   |     1907.00   |
                                 33_| 08/14/1933 |    46500.00   |     1000.00   |     4220.00   |
                                 34_| 10/18/1942 |    29250.00   |      600.00   |     2340.00   |
                                 35_| 01/19/1946 |    28420.00   |      600.00   |     2274.00   |
                                 36_| 01/05/1951 |    24680.00   |      500.00   |     1974.00   |
                                 37_| 03/19/1948 |    29840.00   |      600.00   |     2387.00   |
                                 38_| 03/31/1954 |    28760.00   |      600.00   |     2301.00   |
                                 39_| 03/28/1936 |    26250.00   |      500.00   |     2100.00   |
                                 40_| 04/21/1931 |    15900.00   |      300.00   |     1272.00   |
                                 41_| 07/18/1941 |    25370.00   |      500.00   |     2030.00   |
                                 42_| 05/17/1926 |    23840.00   |      500.00   |     1907.00   |
                                     ---------------------------------------------------------+
SUCCESSFUL RETRIEVAL OF       42 ROW(S)
```

Figure 29 (Part 2 of 2). DSNTEP2 SYSPRINT Output. Note that null values are marked with a question mark (?) in the listing.

6.3.4 Loading the Files into DB2/6000

Before you can load the data into your database you must separately create the DB2 tables. For this you have to retype the create statements or transfer them from your mainframe to your workstation in text mode and run them through the DB2 program (see Figure 30).

```
#!/bin/ksh
db2 CONNECT TO yourdb
db2 -tf /your/create/table/file
db2 -tf /your/create/table/file2
db2 CONNECT reset
```

Figure 30. Sample Create Table in DB2/6000

After you have created these DB2 tables you can load the data. The script shown in Figure 31 on page 34 loads data files in which the columns are separated by delimiters.

```
#!/bin/ksh
db2 CONNECT TO yourdb
db2 IMPORT FROM /your/unload/file OF DEL MESSAGES msgs.txt \
INSERT INTO your.table
db2 COMMIT
db2 CONNECT RESET
```

Figure 31. Script for Loading DB2 Tables

Run the script for loading DB2 tables from a userid that is authorized to load your DB2/6000 table in your database. The fields in the data file must be separated by a comma, which is the default field delimiter.

Chapter 7. Migrating DB2 Objects

In this chapter we discuss the work involved in preparing and executing your SQL DDL for DB2/6000. Your application will usually consist of the following DB2 objects:

- Databases
- Storage groups
- Tablespaces
- Tables
- Views
- Indexes
- Grants
- Synonyms
- Procedures
- Plans.

There are some slight differences in the SQL statements supported by DB2 MVS and DB2/6000. You can find the differences for your application by consulting the SQL reference guides for both systems or by studying the *Formal Register of Extensions and Differences in SQL* manual.

7.1 Executing SQL Create Statements

DB2/6000 offers a variety of ways to execute SQL statements. You can use the db2 command to connect to the required DB2/6000 database and enter SQL statements interactively. You can also put your SQL statements in an AIX file and invoke the db2 command with the -f flag to tell it to execute the statements in the file. If you have SQL statements that spread over multiple lines, you must have a semicolon (;) at the end of each SQL statement and invoke the db2 command with the -t flag (see Figure 32 and Figure 33 on page 36).

```
CREATE VIEW DSN8310.VDEPT
       AS SELECT ALL      DEPTNO  ,
                          DEPTNAME,
                          MGRNO   ,
                          ADMRDEPT
       FROM DSN8310.DEPT ;

CREATE VIEW DSN8310.VHDEPT
       AS SELECT ALL      DEPTNO  ,
                          DEPTNAME,
                          MGRNO   ,
                          ADMRDEPT,
                          LOCATION
       FROM DSN8310.DEPT ;
```

Figure 32. SQL Statements in AIX File Spread over Multiple Lines

```
#!/bin/ksh
db2 CONNECT TO yourdb
db2 -tf /aix/your/SQL.statements
db2 COMMIT
db2 CONNECT RESET
```

Figure 33. Sample Script to Invoke the db2 Command

You need to run the script to invoke the db2 command from a userid that has the necessary DB2/6000 authorizations to execute the SQL statements.

Alternatively you can execute SQL statements through the IBM program product, QUERY/6000. This product also offers many other features to help you and your users access and maintain the data in your DB2/6000 database.

7.2 Database Definitions

You do not need any of your DB2 MVS database definitions for DB2/6000. The concept of a database is somewhat different in DB2/6000. To retrieve data from DB2/6000 an application has to be connected to a database. Only one connection can be held. To use a second database the connection to the first must be reset. Each DB2/6000 database contains its own set of catalog tables. Users must know to which database they want to connect, as the same table can be in multiple databases. A database belongs to a DB2/6000 instance.

In DB2 MVS a DB2 subsystem is a better analogy to the DB2/6000 database. Because a DB2 MVS database contains all related objects in a logical group, an administrator can start and stop access to those objects with one command.

7.3 Storage Group Definitions

You do not need any of your DB2 MVS storage group definitions for DB2/6000. The concept of storage groups does not exist in DB2/6000. In DB2 MVS objects that are to be stored on the same physical disks are grouped together in storage groups.

7.4 Tablespace Definitions

Version 1.1.0 of DB2/6000, which we used in this project, does not support tablespaces. The announced version 2.1 of DB2/6000 will introduce tablespaces. If you are using version 1.1.0 of DB2/6000, you do not need to transfer any of your tablespace definitions.

7.5 Table Definitions

You can use your DB2 MVS create table statements in DB2/6000 without major changes. The main difference is that you do not create a table in a tablespace. Another difference is that you cannot define field edit and validate procedures (EDITPROC and VALIDPROC). To compare a create table statement for DB2 MVS with a create table statement for DB2/6000, see Figure 34 on page 37 and Figure 35 on page 37.

```
CREATE TABLE DSN8310.EMP
(EMPNO     CHAR(6)        NOT NULL,
FIRSTNME  VARCHAR(12)    NOT NULL,
MIDINIT   CHAR(1)        NOT NULL,
LASTNAME  VARCHAR(15)    NOT NULL,
WORKDEPT  CHAR(3)                ,
PHONENO   CHAR(4)                ,
HIREDATE  DATE                   ,
JOB       CHAR(8)                ,
EDLEVEL   SMALLINT               ,
SEX       CHAR(1)                ,
BIRTHDATE DATE                   ,
SALARY    DECIMAL(9, 2)          ,
BONUS     DECIMAL(9, 2)          ,
COMM      DECIMAL(9, 2)          ,
PRIMARY KEY(EMPNO),
FOREIGN KEY RED (WORKDEPT) REFERENCES DSN8310.DEPT
ON DELETE SET NULL)
EDITPROC  DSN8EAE1
IN DSN8D31A.DSN8S31E;
```

Figure 34. DB2 MVS Create Table Statement

```
CREATE TABLE DSN8310.EMP
(EMPNO     CHAR(6)        NOT NULL,
FIRSTNME  VARCHAR(12)    NOT NULL,
MIDINIT   CHAR(1)        NOT NULL,
LASTNAME  VARCHAR(15)    NOT NULL,
WORKDEPT  CHAR(3)                ,
PHONENO   CHAR(4)                ,
HIREDATE  DATE                   ,
JOB       CHAR(8)                ,
EDLEVEL   SMALLINT               ,
SEX       CHAR(1)                ,
BIRTHDATE DATE                   ,
SALARY    DECIMAL(9, 2)          ,
BONUS     DECIMAL(9, 2)          ,
COMM      DECIMAL(9, 2)          ,
PRIMARY KEY(EMPNO),
FOREIGN KEY RED (WORKDEPT) REFERENCES DSN8310.DEPT
ON DELETE SET NULL);
```

Figure 35. DB2/6000 Create Table Statement

7.6 View Definitions

Apart from any necessary renaming actions you do not have to change your view definitions. You can execute the SQL create statements for views from DB2 MVS in DB2/6000.

7.7 Index Definitions

The index definitions in DB2/6000 do not require any of the storage specific parameters that you define for DB2 MVS indexes. You can shorten the index definition to the closing bracket after the last column. Figure 36 on page 38 and Figure 37 on page 38 show you an existing index definition for DB2 MVS that has been changed to be used with DB2/6000.

```
CREATE UNIQUE INDEX DSN8310.XDEPT1
ON DSN8310.DEPT
(DEPTNO   ASC)
USING STOGROUP DSN8G310
PRIQTY 12
ERASE NO
SUBPAGES 8
BUFFERPOOL BP0
CLOSE NO
DSETPASS DSN8 ;
```

Figure 36. DB2 MVS Create Index Statement

```
CREATE UNIQUE INDEX DSN8310.XDEPT1
ON DSN8310.DEPT
(DEPTNO   ASC);
```

Figure 37. DB2/6000 Create Index Statement

7.8 Grant Definitions

If you use userids on your AIX system that differ from those on your MVS system, you have to change your SQL grant statements to correspond to these new userids.

DB2 MVS has grant statements for the following objects:

- Database privileges
- Plan privileges
- System privileges
- Table or view privileges
- Use privileges.

DB2/6000 has grant statements for the following objects:

- Index privileges
- Database authorities
- Package privileges
- Table or view privileges.

The plan privileges in DB2 MVS correspond to the package privileges in DB2/6000, and the system privileges correspond to the database authorities.

7.9 Synonym Definitions

You cannot define any synonyms in DB2/6000. This should not normally cause a problem for you. If your views and tables have the same name as your synonyms, you just have to create your views and tables using the same qualifier as the userid under which you access your DB2/6000 database. If your views and tables have different names, you have to change your application programs.

7.10 Procedure Definitions

DB2/6000 supports neither EDITPROC nor VALIDPROC on the create table statement. If you use these functions, you have to change your create table statements as shown in 7.5, “Table Definitions” on page 36. There is currently no way of maintaining these functions in DB2/6000 other than providing this function in your application program code.

7.11 Plans and Packages

DB2/6000 uses packages rather than plans to store information in the database needed to execute the SQL statements from a single source file. Packages can be created either during precompiling or through binding a bind file. In DB2 MVS the precompiler produces a database request module (DBRM) for each source file. The bind process groups the DBRMs into a plan.

Once you have transferred all of your application programs to your AIX system you can recompile them and, if required, bind your bind files to create packages. This is discussed in Chapter 9, “Migrating COBOL Programs” on page 53. Afterwards you can change your grant statements for your plans to grant statements for your newly created packages and execute them.

Chapter 8. Migrating Maps

In this chapter we outline the BMS support that CICS/6000 offers and review the steps required to compile and install maps in CICS/6000.

8.1 BMS Support

The existing maps you migrate from other CICS family members are processed within the CICS/6000 environment with the limitation that only minimum BMS functions are supported.

8.1.1 Functions Supported

Minimum function BMS supports the IBM 3270 and 3270-like range of displays and printers (except SCS printers). CICS/6000 supports the following minimum BMS functions:

- Basic 3270 displays and printers: 3270, 3270P, LUTYPE2, LUTYPE3
- Extended attributes
- Formfeed control
- RECEIVE MAP and RECEIVE MAP FROM
- Map set suffixing
- Aligned and unaligned maps
- Out of sequence input maps
- Block data
- Automatic setting of WCC character line width
- ERASE, ERASEUP, FORMFEED, CURSOR, and WCC on BMS SENDs.

8.1.2 Functions Not Supported

CICS/6000 does not support the following minimum BMS functions:

- Non ACCUM SEND MAP TERMINAL
- Default and alternate screen sizes
- GDDM* coordination

Standard and full function BMS are not supported by CICS/6000, so you will have to modify your applications to overcome these restrictions. There is an example of these modifications in Appendix E, “Overcoming Full Function BMS Restrictions” on page 119.

Standard BMS comprises the following functions:

- Outboard functions
- Partitions
- MSR control
- NLEOM mode for printers
- LUTYPE1 attachment of printers (SCSPRT)
- All other features of all other terminal types
- RECEIVE PARTN
- SEND PARTNSET
- OUTPARTN, ACTPARTN, MSR, LDC, NLEOM, and FMHPARM on SENDs
- INPARTN on RECEIVE MAP.

Full BMS comprises the following functions:

- Non ACCUM SEND TEXT TERMINAL
- Cumulative mapping, floating maps, header and trailer maps
- Page overflow
- Cumulative text
- Terminal operator purging
- CMSG message switching transaction
- ROUTE
- SEND PAGE
- PURGE MESSAGE
- ACCUM, PAGING, SET, and REQID on BMS SENDs
- HEADER, TRAILER, JUSTIFY, JUSLAST, JUSFIRST, and SEND TEXTs
- SEND TEXT MAPPED
- SEND TEXT NOEDIT.

Other restrictions include:

- Display screen size is limited to 24 lines by 80 columns.
- BMS paging and CSPG are not supported. Applications that use the BMS page retrieval function will need to have the function built into the application program logic.
- CICS/6000 ignores the following DFHMDF options:
  - TRANSP
  - OUTLINE
  - VALIDN.

For additional information about BMS functions and levels of support, please refer to Chapters 15 through 18 of the *CICS/ESA Application Programming Guide*, and Chapter 6 of the *AIX CICS/6000 Application Programming Guide*.

8.1.3 Screen Definition Facility II

If you want to migrate map definitions from Screen Definition Facility II (SDF II) to CICS/6000 you need to instruct SDF II to produce BMS definitions. You can download these BMS definitions to the IBM RISC System/6000, but you may experience the following difficulties with maps produced from these definitions:

- SDF II allows field names to be eight characters long, whereas CICS/6000 restricts field names to seven characters.

  You may solve this problem by either:

  - Shortening all field names to seven characters or less and changing all relevant application programs to match the new names

    or

  - Editing your BMS source to modify any field names longer than seven characters and then compiling this source file with the cicsmap -p command. The -p option will produce only a physical map. As no new symbolic map is produced you still use the COBOL copybooks from your previous environment in your application. This is a quick solution to the problem but could lead to maintenance problems as symbolic and physical maps now need to be maintained separately.

- SDF II does not generate DFHMDF labels.

- SDF II allows arrays in logical maps. Such arrays will have to be recoded for CICS/6000.

8.2 Compiling the Maps

CICS/6000 uses the BMS processor to compile the files you have transferred from the host into symbolic and physical map files. The symbolic map file is a programming source language data structure, either a COBOL DATA DIVISION definition or a C structure, depending on the value of LANG in the mapset macro (DFHMSD).

The BMS processor is invoked directly from the command line using the command shown in Figure 38.

```
cicsmap DSN8CCG.bms
```

Figure 38. cicsmap Command

The input file (DSN8CCG) must have an extension of .bms. The processor places the output physical map file and the output symbolic map file in the current working directory. The output physical map has an extension of .map; the symbolic map file has no extension for COBOL. You can use the symbolic map file as a COBOL copybook file by using the COBOL verb COPY.

If the BMS processor detects any errors in the map source file, maps are not generated.

Refer to the CICS/6000 documentation for a full description of the cicsmap command.

8.2.1 Define Maps to CICS/6000 Using SMIT

Once you have processed the map successfully you can move it to your CICS/6000 region whith the command shown in Figure 39.

```
$ cp DSN8CCG.map /var/cics_region/<region>/maps/prime/DSN8CCG.map
```

Figure 39. cp Command

You have to be a member of the cics group to move files into the CICS/6000 region.

Now you need to define CICS/6000 resources by using SMIT as shown in Figure 41 on page 44 or writing a shell script that will add the map as shown in Figure 42 on page 45.

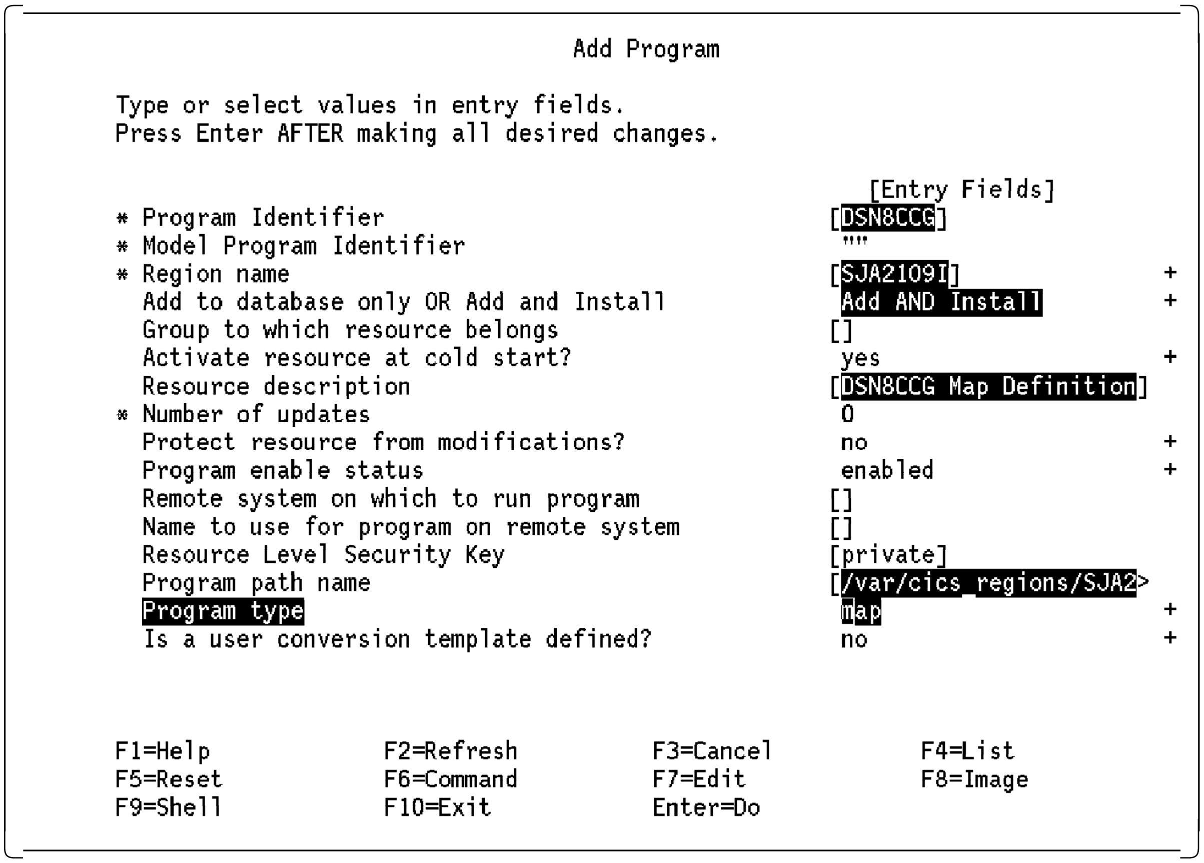
To access and modify CICS/6000 Resource Definitions your AIX user identifier must belong to the cics group.

Enter the SMIT command with the options sequence shown in Figure 40 on page 44 to define the map to CICS/6000, or use the fastpath command `smitty cicsaddpd`.

```
$ smitty cics
   ►Manage CICS/6000 Regions
     ►Define Resources for a CICS/6000 Region
       ►Manage Resources
         ►Programs
           ►Add New
             ►Model Program Identifier
```

Figure 40. SMIT Options Sequence to Define a Map

Select the model ″″ when prompted for a model program identifier. You will now have a screen that looks similar to Figure 41.



```
                              Add Program

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
* Program Identifier                                [DSN8CCG]
* Model Program Identifier                           ""
* Region name                                       [SJA2109I]            +
  Add to database only OR Add and Install            Add AND Install      +
  Group to which resource belongs                   []
  Activate resource at cold start?                   yes                  +
  Resource description                              [DSN8CCG Map Definition]
* Number of updates                                  0
  Protect resource from modifications?               no                   +
  Program enable status                              enabled              +
  Remote system on which to run program             []
  Name to use for program on remote system          []
  Resource Level Security Key                       [private]
  Program path name                                 [/var/cics_regions/SJA2>
  Program type                                       map                  +
  Is a user conversion template defined?             no                   +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 41. Add Program Definition

The full program path name as shown is `/var/cics_regions/SJA2109I/maps/prime/DSN8CCG.map`.

8.2.2 Define Maps to CICS/6000 Using Scripts

You may prefer to write shell scripts to define your maps to CICS/6000 rather than installing each entry individually using SMIT. Figure 42 on page 45 shows a sample shell script for installing the map (DNS8CCG.map).

```
#!/bin/ksh
MAPDIR=/var/cics_regions/$CICSREGION/maps/prime
cp DSN8CCGM.map $MAPDIR
cicsadd -c pd -r $CICSREGION -B ″DSN8CCGM″ \
                 ResourceDescription=″DSN8CCGM Map Definition″ \
                 PathName=″$MAPDIR/DSN8CCGM.map″ \
                 ProgType=map
```

Figure 42. Sample Shell Script to add a CICS/6000 Map Definition

This script adds the definition to both the permanent database and run-time system. You must log in to DCE before attempting to install any definition in the run-time system.

8.2.3 Screen Design Aid

Screen Design Aid (SDA) is a GUI tool supplied with CICS/6000 that allows you to design and edit BMS screens (new or existing) interactively using the AIXWindows environment.

8.3 Example

For purposes of comparison a BMS source file is shown in Figure 43 on page 46; the COBOL copybook file as generated on MVS, in Figure 44 on page 48; and the COBOL copybook file generated by the BMS processor in CICS/6000, in Figure 45 on page 50. Even though the COBOL structures are not defined in the same manner, the corresponding names refer to the same actual storage locations.

```
DSN8CCG DFHMSD TYPE=MAP,
               MODE=INOUT,
               LANG=COBOL,
               TERM=3270-2,
               CTRL=(FREEKB,FRSET),
               STORAGE=AUTO,
               TIOAPFX=YES
DSN8CCG DFHMDI SIZE=(24,80),
               LINE=1,
               COLUMN=1
ATITLE  DFHMDF POS=(1,15),
               ATTRB=PROT,
               LENGTH=50
        DFHMDF POS=(2,1),
               ATTRB=PROT,
               INITIAL='MAJOR SYSTEM ...:',
               LENGTH=17
AMAJSYS DFHMDF POS=(2,19),
               ATTRB=(PROT,BRT,FSET),
               INITIAL='O',
               LENGTH=1
        DFHMDF POS=(2,21),
               ATTRB=ASKIP,
               LENGTH=1
        DFHMDF POS=(2,30),
               ATTRB=PROT,
               INITIAL='ORGANIZATION     ',
               LENGTH=50
        DFHMDF POS=(3,1),
               ATTRB=PROT,
               INITIAL='ACTION .........:',
               LENGTH=17
AACTION DFHMDF POS=(3,19),
               ATTRB=(UNPROT,BRT,IC,FSET),
               LENGTH=1
        DFHMDF POS=(3,21),
               ATTRB=ASKIP,
               LENGTH=1
ADESCL2 DFHMDF POS=(3,30),
               ATTRB=PROT,
               LENGTH=50
        DFHMDF POS=(4,1),
               ATTRB=PROT,
               INITIAL='OBJECT .........:',
               LENGTH=17
```

Figure 43 (Part 1 of 2). Original BMS Source Code

```
AOBJECT DFHMDF POS=(4,19),
               ATTRB=(UNPROT,BRT,FSET),
               LENGTH=2
        DFHMDF POS=(4,22),
               ATTRB=ASKIP,
               LENGTH=1
ADESCL3 DFHMDF POS=(4,30),
               ATTRB=PROT,
               LENGTH=50
        DFHMDF POS=(5,1),
               ATTRB=PROT,
               INITIAL='SEARCH CRITERIA.:',
               LENGTH=17
ASEARCH DFHMDF POS=(5,19),
               ATTRB=(UNPROT,BRT,FSET),
               LENGTH=2
        DFHMDF POS=(5,22),
               ATTRB=ASKIP,
               LENGTH=1
ADESCL4 DFHMDF POS=(5,30),
               ATTRB=PROT,
               LENGTH=50
        DFHMDF POS=(6,1),
               ATTRB=PROT,
               INITIAL='DATA ...........:',
               LENGTH=17
ADATA   DFHMDF POS=(6,19),
               ATTRB=(UNPROT,BRT,FSET),
               LENGTH=60
AMSG    DFHMDF POS=(7,1),
               ATTRB=(PROT,BRT),
               LENGTH=79
ALINE   DFHMDF POS=(9,1),
               ATTRB=PROT,
               LENGTH=79,
               OCCURS=15
APFKEY  DFHMDF POS=(24,1),
               ATTRB=PROT,
               LENGTH=79
        DFHMSD TYPE=FINAL
         END
```

Figure 43 (Part 2 of 2). Original BMS Source Code

```
       01  DSN8CCGI.
           02  FILLER PIC X(12).
           02  ATITLEL    COMP  PIC  S9(4).
           02  ATITLEF    PICTURE X.
           02  FILLER REDEFINES ATITLEF.
             03 ATITLEA    PICTURE X.
           02  ATITLEI  PIC X(50).
           02  AMAJSYSL    COMP  PIC  S9(4).
           02  AMAJSYSF    PICTURE X.
           02  FILLER REDEFINES AMAJSYSF.
             03 AMAJSYSA    PICTURE X.
           02  AMAJSYSI  PIC X(1).
           02  AACTIONL    COMP  PIC  S9(4).
           02  AACTIONF    PICTURE X.
           02  FILLER REDEFINES AACTIONF.
             03 AACTIONA    PICTURE X.
           02  AACTIONI  PIC X(1).
           02  ADESCL2L    COMP  PIC  S9(4).
           02  ADESCL2F    PICTURE X.
           02  FILLER REDEFINES ADESCL2F.
             03 ADESCL2A    PICTURE X.
           02  ADESCL2I  PIC X(50).
           02  AOBJECTL    COMP  PIC  S9(4).
           02  AOBJECTF    PICTURE X.
           02  FILLER REDEFINES AOBJECTF.
             03 AOBJECTA    PICTURE X.
           02  AOBJECTI  PIC X(2).
           02  ADESCL3L    COMP  PIC  S9(4).
           02  ADESCL3F    PICTURE X.
           02  FILLER REDEFINES ADESCL3F.
             03 ADESCL3A    PICTURE X.
           02  ADESCL3I  PIC X(50).
           02  ASEARCHL    COMP  PIC  S9(4).
           02  ASEARCHF    PICTURE X.
           02  FILLER REDEFINES ASEARCHF.
             03 ASEARCHA    PICTURE X.
           02  ASEARCHI  PIC X(2).
           02  ADESCL4L    COMP  PIC  S9(4).
           02  ADESCL4F    PICTURE X.
           02  FILLER REDEFINES ADESCL4F.
             03 ADESCL4A    PICTURE X.
           02  ADESCL4I  PIC X(50).
           02  ADATAL    COMP  PIC  S9(4).
           02  ADATAF    PICTURE X.
           02  FILLER REDEFINES ADATAF.
             03 ADATAA    PICTURE X.
           02  ADATAI  PIC X(60).
           02  AMSGL    COMP  PIC  S9(4).
           02  AMSGF    PICTURE X.
           02  FILLER REDEFINES AMSGF.
             03 AMSGA    PICTURE X.
           02  AMSGI  PIC X(79).
           02  ALINED OCCURS 15 TIMES.
             03  ALINEL    COMP  PIC  S9(4).
             03  ALINEF    PICTURE X.
             03  ALINEI  PIC X(79).
           02  APFKEYL    COMP  PIC  S9(4).
           02  APFKEYF    PICTURE X.
           02  FILLER REDEFINES APFKEYF.
             03 APFKEYA    PICTURE X.
           02  APFKEYI  PIC X(79).
```

Figure 44 (Part 1 of 2). MVS CICS COBOL Copybook

```
       01  DSN8CCGO REDEFINES DSN8CCGI.
           02  FILLER PIC X(12).
           02  FILLER PICTURE X(3).
           02  ATITLEO  PIC X(50).
           02  FILLER PICTURE X(3).
           02  AMAJSYSO  PIC X(1).
           02  FILLER PICTURE X(3).
           02  AACTIONO  PIC X(1).
           02  FILLER PICTURE X(3).
           02  ADESCL2O  PIC X(50).
           02  FILLER PICTURE X(3).
           02  AOBJECTO  PIC X(2).
           02  FILLER PICTURE X(3).
           02  ADESCL3O  PIC X(50).
           02  FILLER PICTURE X(3).
           02  ASEARCHO  PIC X(2).
           02  FILLER PICTURE X(3).
           02  ADESCL4O  PIC X(50).
           02  FILLER PICTURE X(3).
           02  ADATAO  PIC X(60).
           02  FILLER PICTURE X(3).
           02  AMSGO  PIC X(79).
           02  DFHMS1 OCCURS 15 TIMES.
             03  FILLER PICTURE X(2).
             03  ALINEA    PICTURE X.
             03  ALINEO  PIC X(79).
           02  FILLER PICTURE X(3).
           02  APFKEYO  PIC X(79).
```

Figure 44 (Part 2 of 2). MVS CICS COBOL Copybook

```
*
* cicsmap DSN8CCG.bms -- Thu Jun 16 14:16:52 PDT 1994
* DSN8CCG DFHMSD MODE=INOUT,STORAGE=AUTO
*
 01  DSN8CCGI.
     02   FILLER    PIC X(12).
     02   ATITLEL   PIC S9(4) COMP.
     02   ATITLEF   PIC X.
     02   ATITLEI   PIC X(50).
     02   AMAJSYSL  PIC S9(4) COMP.
     02   AMAJSYSF  PIC X.
     02   AMAJSYSI  PIC X(1).
     02   AACTIONL  PIC S9(4) COMP.
     02   AACTIONF  PIC X.
     02   AACTIONI  PIC X(1).
     02   ADESCL2L  PIC S9(4) COMP.
     02   ADESCL2F  PIC X.
     02   ADESCL2I  PIC X(50).
     02   AOBJECTL  PIC S9(4) COMP.
     02   AOBJECTF  PIC X.
     02   AOBJECTI  PIC X(2).
     02   ADESCL3L  PIC S9(4) COMP.
     02   ADESCL3F  PIC X.
     02   ADESCL3I  PIC X(50).
     02   ASEARCHL  PIC S9(4) COMP.
     02   ASEARCHF  PIC X.
     02   ASEARCHI  PIC X(2).
     02   ADESCL4L  PIC S9(4) COMP.
     02   ADESCL4F  PIC X.
     02   ADESCL4I  PIC X(50).
     02   ADATAL    PIC S9(4) COMP.
     02   ADATAF    PIC X.
     02   ADATAI    PIC X(60).
     02   AMSGL     PIC S9(4) COMP.
     02   AMSGF     PIC X.
     02   AMSGI     PIC X(79).
     02   ALINED OCCURS 15 TIMES.
       03 ALINEL    PIC S9(4) COMP.
       03 ALINEF    PIC X.
       03 ALINEI    PIC X(79).
     02   APFKEYL   PIC S9(4) COMP.
     02   APFKEYF   PIC X.
     02   APFKEYI   PIC X(79).
```

Figure 45 (Part 1 of 2). CICS/6000 COBOL Copybook

```
       01  DSN8CCGO REDEFINES DSN8CCGI.
           02   FILLER    PIC X(12).
           02   FILLER    PIC S9(4) COMP.
           02   ATITLEA   PIC X.
           02   ATITLEO   PIC X(50).
           02   FILLER    PIC S9(4) COMP.
           02   AMAJSYSA  PIC X.
           02   AMAJSYSO  PIC X(1).
           02   FILLER    PIC S9(4) COMP.
           02   AACTIONA  PIC X.
           02   AACTIONO  PIC X(1).
           02   FILLER    PIC S9(4) COMP.
           02   ADESCL2A  PIC X.
           02   ADESCL2O  PIC X(50).
           02   FILLER    PIC S9(4) COMP.
           02   AOBJECTA  PIC X.
           02   AOBJECTO  PIC X(2).
           02   FILLER    PIC S9(4) COMP.
           02   ADESCL3A  PIC X.
           02   ADESCL3O  PIC X(50).
           02   FILLER    PIC S9(4) COMP.
           02   ASEARCHA  PIC X.
           02   ASEARCHO  PIC X(2).
           02   FILLER    PIC S9(4) COMP.
           02   ADESCL4A  PIC X.
           02   ADESCL4O  PIC X(50).
           02   FILLER    PIC S9(4) COMP.
           02   ADATAA    PIC X.
           02   ADATAO    PIC X(60).
           02   FILLER    PIC S9(4) COMP.
           02   AMSGA     PIC X.
           02   AMSGO     PIC X(79).
           02   DFHMS01 OCCURS 15 TIMES.
             03 FILLER    PIC S9(4) COMP.
             03 ALINEA    PIC X.
             03 ALINEO    PIC X(79).
           02   FILLER    PIC S9(4) COMP.
           02   APFKEYA   PIC X.
           02   APFKEYO   PIC X(79).
```

Figure 45 (Part 2 of 2). CICS/6000 COBOL Copybook

Chapter 9. Migrating COBOL Programs

In this chapter we provide information about the migration of COBOL applications that use both CICS and DB2 to the AIX environment.

Before migrating COBOL programs to CICS/6000 and DB2/6000, you should review your CICS application and system programs, together with any other CICS-related programs, to determine their level of compatibility with CICS/6000. Some programs may need to be modified substantially or even redesigned to make them compliant with CICS/6000.

9.1 Source Language and Compiler Considerations

CICS/6000 does not support any source languages other than COBOL and C. Applications written in another language will need to be recoded for porting purposes. Assembler routines are encountered fairly frequently in the mainframe CICS environment. These must be rewritten in COBOL or C for CICS/6000. Tools are available for the porting of languages other than COBOL or C. These tools are code converters and convert approximately 60% of code; the remainder has to be done manually. The conversions available are:

- PLI ► C
- COBOL ► C
- ASM370 ► C
- ASM370 ► COBOL.

Refer to Appendix A, “Migration Tidbits” on page 97 for the availability of such tools.

9.1.1 COBOL Migration Restrictions

The following restrictions apply to COBOL when used in the CICS/6000 environment:

- COBOL base locator linkage (BLL) cells

  COBOL does not support BLL cells. If the existing application accesses BLL cells, it will have to be rewritten.

- Service reload statements

  Service reload statements are not supported by COBOL currently; applications containing service reload statements will have to be rewritten.

- COBOL copybooks

  The CICS/6000 translator (cicstran) does not currently handle EXEC CICS statements in COBOL copybooks. Therefore, if your current COBOL copybooks include such statements, they will have to be changed accordingly. Copybooks that do not include CICS calls are resolved correctly.

  The CA60 SupportPac offers a tool called cpyHandle. This tool flattens the copybooks provided with a CICS COBOL program should they contain CICS calls. cpyHandle can be run as part of the make process, so that the program and copybook structure separation can be maintained. See A.5, “Hursley SupportPacs” on page 99.

- Nested COBOL programs

  The CICS/6000 translator does not currently handle the nesting feature of COBOL. Applications that use this feature will have to be modified accordingly.

9.1.2 CICS/6000 Migration Restrictions

The following restrictions must be observed when migrating a CICS application to CICS/6000:

- Macro-level applications

  CICS/6000 does not support macro-level application programs. If your existing CICS application contains macro-level statements, you will have to recode them to use the API commands that CICS/6000 supports.

  The CICS Application Migration Aid (AMA) assists in the conversion from macro-level to command-level CICS. This tool scans code for macro statements. When it finds one, it attempts to replace the macro with the equivalent API command. If this is not possible, it indicates what needs to be done to replace the macro statement. Refer to Appendix A, "Migration Tidbits" on page 97 for the availability of this tool.

- Command API Restrictions

  The *CICS Family: API Structure* manual documents the differences between each CICS platform API. This manual is designed to help you develop portable applications that can run on more than one CICS platform and should be reviewed as part of your application migration planning.

9.1.3 Setting Up the COBOL Run-Time System

To configure CICS/6000 to support DB2/6000 transactions, you must run the `cicsmkcobol` script. This script creates and installs a file called `cicsprCOBOL` in the /usr/lpp/cics/v1.1/bin directory, which contains the COBOL support routines and run-time information. Before running `cicsmkcobol`, you must set the COBDIR environment variable to the directory path where your COBOL resides and your PATH environment variable to include the COBOL bin directory.

You should perform the following steps as user root:

1. Change to the appropriate directory by entering the command shown in Figure 46.

```
cd /usr/lpp/cics/v1.1/bin
```

Figure 46. Change Directory

2. Run `cicsmkcobol`.

If you created a symbolic link as shown in B.1.1, "Shared Object" on page 103, enter the command as shown in Figure 47.

```
cicsmkcobol /usr/lib/db2.o
```

Figure 47. cicsmkcobol Command

Otherwise issue the command as shown in Figure 48 on page 55.

```
cicsmkcobol -L/usr/lpp/db2_01_01_0000/lib
                   /usr/lpp/db2_01_01_0000/lib/lib/db2.o
```

Figure 48. cicsmkcobol Command (No Links)

When run with this syntax the *cicsmkcobol* builds a COBOL run-time environment with CICS/6000 and DB2/6000 support.

9.1.4 Compiling COBOL Programs

The following steps are needed to compile CICS/6000 COBOL programs that use DB2/6000 in an XA environment:

1. Ensure that your COBDIR and COBCPY environment variables are set correctly.

   Your COBCPY environment variable must contain the CICS/6000 and DB2/6000 include directories, as well as your application include directory.

2. Pass the source through the CICS/6000 translator (cicstran).

3. Use COBOL to precompile, compile, and link the resulting file.

Figure 49 shows a sample makefile used during this project.

Note

To execute the makefile you must have DB2/6000 authorization to perform all SQL statements embedded in the source COBOL program and BINDADD privilege for the specific database. This will enable you to create a new database package.

```
DSN8CC2.gnt:
    rm -f  DSN8CC2.cbl DSN8CC2.gnt DSN8CC2.int
    cicstran -e -d -lCOBOL DSN8CC2
    COBCPY=/usr/lpp/cics/v1.1/include:/usr/lpp/db2_01_01_0000/include: \
           /project/include
        export COBCPY
        cob -P -A -v -C NOSQLINIT \
                -C SQLDB2 \
                -C SQLNOT'"94"' \
                -C SQLDB'"sample"'\
                -C early-release\
                -C SQLBIND'""' \
                -C VSC2 DSN8CC2.cbl
    db2 connect to sample; \
    db2 bind DSN8CC2.bnd isolation cs
    db2 grant execute on package DSN8CC2 to public;
```

Figure 49. Makefile for DSN8CC2. Depending on the release of your COBOL compiler, you may not need to specify the early-release compiler directive. The release we used required the early-release directive to process imbedded SQL statements.

9.1.5 Problems Encountered

We encountered the following problems during the migration of the COBOL application programs to the CICS/6000 and DB2/6000 environments:

- COBOL copybook not found due to case sensitivity

  A COBOL program transferred from the mainframe contains this statement:

  ```
  EXEC SQL INCLUDE DSN8MCC2 END-EXEC.
  ```

  where DSN8MCC2 is a COBOL copybook. When transferred from the mainframe the copybook name remained in uppercase, and the SQL INCLUDE statement could not find this copybook until the name of the copybook was changed to lowercase in the AIX directory. However, all copybook file names included using the COBOL COPY statement, for example, `COPY DSN8MCC2`, must be in uppercase.

- Map attributes

  This mainframe application dynamically changes maps and does not use the standard CICS constants to set attribute values such as ′protected′ but uses hard-coded numeric values:

  ```
  MOVE +225 TO ATTR2(I).
  ```

  The application program logic uses these values in the attribute bytes to determine which fields to select from the DB2 database. This practice does not make the application easy to port between CICS platforms.

- Reserved word cannot be used

  You cannot use any reserved COBOL words as a user-defined word, for example, as a variable name, or as a label name. You can use a reserved word as a user-defined word in MVS VS COBOL II, so you may have to rename your user-defined word to compile your program on the AIX system.

- Different keywords for COBOL statements

  Certain keywords are not the same in the COBOL for UNIX compiler and the MVS COBOL VS II compiler. Normally, you can refrain from using these keywords.

9.2 Database and SQL Differences

In this section we examine the differences between DB2/6000 and DB2 MVS database restrictions. We also briefly discuss the differences between the DB2/6000 and DB2 MVS precompilers. As this book focuses on CICS applications, the call attachment facility (CAF) is not covered. Our application uses embedded static SQL.

9.2.1 Restrictions

Table 3 on page 57 compares the most important restrictions between DB2 MVS and DB2/6000 version 1.1. The announced version 2.1 of DB2/6000 provides higher limits for some of these restrictions.

Table 3. Restrictions for DB2 MVS and DB2/6000

Feature	**DB2 MVS**	**DB2/6000**
Longest host identifier	64	30
Maximum length for VARCHAR	32704	4000
Maximum length for VARGRAPHIC	16352	2000
Maximum number of columns in a table	750	255
Maximum number of columns in a view	750	255
Maximum number of columns in an index key	64	16
Maximum number of host variables in a program	available storage	880

Other limits and restrictions are listed in Appendix A, "SQL Limits," of the *DATABASE 2 AIX/6000 and DATABASE 2 OS/2 SQL Reference*. If your application exceeds DB2/6000 limits but not DB2 MVS limits, you have to rework it. A complete list of the SQL differences among all DB2 systems is provided in the *Formal Register of Extensions and Differences in SQL* manual.

Note

The maximum number of 15 tables in an SQL statement includes, for each view, the view and all tables on which the view is based in DB2/6000. In DB2 MVS only the tables are included towards this limit. This can cause compilation problems on AIX for programs that run on MVS. Normally you can resolve this by referring directly to the tables in your SQL statement instead of the views. This limit has been removed in version 2.1 of DB2/6000.

9.2.2 Precompiler

A few minor differences exist between the DB2/6000 and the DB2 MVS precompiler. Each host variable must be preceded by a colon in DB2/6000. You may not have any blanks between the host variable and the preceding colon.

DB2/6000 does not allow any hyphens in cursor names. On DB2 MVS this is allowed.

The *not* operator is not the same on all platforms. You should use the word *not* or <>. This is completely portable. If you do not want to change your programs and you use the *not* operator, you can set the SQLNOT compiler directive of your COBOL compiler to your *not* operator (see Figure 49 on page 55).

The COBOL compiler allows you to increase the compatibility between DB2/6000 and DB2 MVS programs. By specifying the compiler directive, SQLDB2, you do not have to have the declarations of all of your host variables within an SQL declare section, which is also not mandatory in MVS (see Figure 49 on page 55).

The DB2 precompiler is integrated into the COBOL compiler. You do not have to separately prepare and bind your program with embedded SQL statements.

9.2.3 DSNTIAR

DSNTIAR is the commonly used DB2 MVS program to retrieve and format error messages for a specific SQL error code. In DB2/6000 you must use sqlgintp to retrieve and format your SQL error message. The *DATABASE 2 AIX/6000 Programming Reference* describes sqlgintp. If you are programming in C, you have to use sqlaintp instead of sqlgintp.

9.3 Testing Your Application

Once you have successfully compiled your programs you need to test them. If you do not receive the expected results or you want to perform a glassbox test, you can use the CICS or COBOL facilities to trace your transactions. In this section we show you how to set up your environment and how to compile your applications to trace them.

9.3.1 Execution Diagnostic Facility

You use the CICS/6000-supplied CEDF transaction to control the Execution Diagnostic Facility (EDF). EDF enables you to debug an application program without modifying the program. CICS/6000 passes control to EDF at specific interception points. EDF displays the state of the application program at the interception point. It also allows you to interact with EDF screens to view additional information about the program and overtype certain areas of the screen to test the execution of the program before returning control to the application. Additional information about the EDF can be found in the *AIX CICS/6000 CICS-Supplied Transactions* manual.

9.3.1.1 Translating Your Application Program

To use CEDF you must specify the -e flag when running the CICS/6000 translator, cicstran (see Figure 50).

```
cicstran -e -d -lCOBOL YOURPGM
```

Figure 50. Running cicstran for CEDF

If the -d flag of the cicstran command is also used, the translator source listing has line numbers that EDF can use. EDF can then identify the command by line number. For additional information, see “cicstran Command” in Chapter 31, “Application Development Commands and Procedures,” in the *CICS/6000 Application Programming Guide*.

9.3.1.2 Setting up Security for CEDF

The TSLCheck value in the transaction definitions (TD) entry must be the same for the CEDF and your application′s transactions. The TSLKey values in the TD of both transactions must match. If any of the values do not match, you cannot run CEDF, and a NOTAUTH condition is raised. For additional information, see “Authorizing Access to the CEDF Debugging Transaction” in the *CICS/6000 Customization and Operation* manual.

9.3.1.3 Running CEDF

You must run EDF on a different terminal from the terminal on which the transaction is to be tested. To do this start two CICS terminals. You can then invoke the CEMT transaction on one terminal to find the four-character terminal identifier of the current terminal. Figure 51 shows you how to invoke the CEMT transaction with the correct arguments.

```
CEMT I TE
```

Figure 51. CEMT Inquire Terminals

Figure 52 shows the output of CEMT.

```
I TE
STATUS:  RESULTS - OVERTYPE TO MODIFY

  Ter(AE01)  Trn(CEMT) Pri( 000 ) Ins Ati Tti
          Net(AEGETS01)
  Ter(PRT2)  Trn(    ) Pri( 000 ) Ins Ati Tti
          Net(PRINTER2)















                                                           APPLID=SJA2109I
RESPONSE:  NORMAL
 PF 1 HELP      3 END                   7 SBH 8 SFH 9 MSG 10 SB 11 SF
```

Figure 52. CEMT Output

Your terminal identifier is the identifier corresponding to the CEMT transaction. In this case it is AE01. You can now start your EDF session by invoking the CEDF transaction with your terminal identifier as the argument (see Figure 53).

```
CEDF AE01
```

Figure 53. Invoking CEDF

You must invoke CEDF from your other terminal, not from the AE01 terminal. The message “Terminal AE01: EDF mode ON” shows you that you have started CEDF correctly. You can now run your transaction from the AE01 terminal. At each CICS statement the program execution will be intercepted and control will be passed to EDF.

9.3.2 COBOL Source Level Debugger

COBOL provides you with a source level debugger for debugging CICS applications. You can view your COBOL source code while the program is executing. You can single-step through the program or let it run in slow motion. You can interrupt the execution by setting breakpoints, and you can monitor data items.

9.3.2.1 Compiling Your Program for Debugging

You must compile all of the COBOL programs you want to debug with the -A flag as shown in Figure 54.

```
cob -P -A -v YOURPGM.cbl
```

Figure 54. Compiling COBOL Programs for Debugging

When you use the -A flag to compile your program, YOURPGM.cbl, three files are generated:

- YOURPGM.gnt
- YOURPGM.int
- YOURPGM.idy.

You must now copy the YOURPGM.int and YOURPGM.idy files and the program source code file, YOURPGM.cbl, to the /var/cics_regions/YOURREGION/bin directory.

9.3.2.2 Running Your Program with Source Level Debugger

To test a CICS application at source level you must start an aixterm to display the program source code (see Figure 55). This will start a terminal that will be used later to display the program source code. In the top left-hand corner of this terminal you can see its name. The name should be something like /dev/pts/4.

```
aixterm -e ksh -c ′tty; exec sleep 99999′ &
```

Figure 55. Starting an aixterm for Debugging

You must now make this terminal accessible to the debugging facilities. Issue the chmod command as shown in Figure 56.

```
chmod 666 /dev/pts/4
```

Figure 56. Executing chmod for /dev/pts/4

You can now start your cicsterm with the -A flag to invoke the debugger. The argument for the -A flag must be the tty name of your aixterm, for example, /dev/pts/4 (see Figure 57).

```
cicsterm -r YOURREGION -A /dev/pts/4
```

Figure 57. Starting a cicsterm for Debugging

You can now invoke your transaction at this cicsterm, and the debugger output is sent to your aixterm.

9.3.3 COBOL Program Caching

To improve performance the COBOL run-time system (RTS) caches programs in memory and uses those copies rather than a new copy from disk for the next invocation. CICS/6000 cannot influence this or force a new copy of a program to be loaded. If you want to have a new copy of your program loaded for each invocation, for example, for your development environment, you must set the COBOL environment variable, COBSW to -l0, in the environment file of your CICS region (/var/cics_regions/YOURREGION/environment) and restart your CICS region (see Figure 58 on page 62).

```
#
# NAME:         environment - Environment file for CICS
#
# VERSION:      1.1
#
# DESCRIPTION:
#       CICS environment file. This file should only contain
#           1.  comment lines which have a # in the first column,
#           2.  blank lines, and
#           3.  Lines in the form name=value.
#
#       For example, if you wanted to alter the value of the LANG
#       environment variable so that the region runs with LANG set
#       to Ja_JP, you would add the line :-
#
#           LANG=Ja_JP
#
# CAVEATS/WARNINGS:
#       This file is only for establishing environment variables.
#       Execution of commands from this file or any lines other
#       than specified above may cause failure of the initialization
#       process.
#

#-----------------------------------------------------------------------
# Force reload for COBOL programs for each invocation
#-----------------------------------------------------------------------
COBSW=-l0

#-----------------------------------------------------------------------
# Set the DB2INSTANCE environment variable to the name of the instance.
# The name of the instance is the login name of the AIX user that owns
# the instance.
#-----------------------------------------------------------------------
DB2INSTANCE=db2

#-----------------------------------------------------------------------
# DB2BQTIME, DB2BQTRY, DB2RQTIME, DB2IQTIME are used to establish
# the relationship between the front end and back end processes of the
# Command Line Processor.
# It is recommended that you use the default values unless you
# experience problems in your specific environment; if you need to
# change these values, it is not advisable to set them to high.
#-----------------------------------------------------------------------
#-----------------------------------------------------------------------
# DB2BQTIME ·Default= 1 second, Minimum value= 1 second“
# specifies the amount of time the frontend process will sleep before
# checking if the backend process is active and establishing a
# connection to it.
#-----------------------------------------------------------------------
DB2BQTIME=1

#-----------------------------------------------------------------------
# DB2BQTRY ·Default= 60 retries, Minimum value= 0 retries“
# specifies the number of times the frontend process tries to determine
# whether the backend process is already active.
# It works in conjunction with DB2BQTIME.
#-----------------------------------------------------------------------
DB2BQTRY=60
```

Figure 58 (Part 1 of 2). Sample CICS Region Environment File

```
#-----------------------------------------------------------------------
# DB2RQTIME ·Default= 60 seconds, Minimum value= 1 second“
# specifies the amount of time the backend process waits for a request
# from the frontend process.
#-----------------------------------------------------------------------
DB2RQTIME=60

#-----------------------------------------------------------------------
# DB2IQTIME ·Default= 5 seconds, Minimum value= 1 second“
# specifies the amount of time the backend process waits on the input
# queue for the frontend process to pass commands.
#-----------------------------------------------------------------------
DB2IQTIME=5

#-----------------------------------------------------------------------
# DB2DBDFT ·Default= SAMPLE“
# is set to the database alias name of the database that will
# be implicitly connected to when applications are started.
#-----------------------------------------------------------------------
DB2DBDFT=SAMPLE

#-----------------------------------------------------------------------
# DB2COMM ·Default= null, Values: TCPIP APPC NONE“
# specifies which communication protocol will be enabled when the
# database manager is started.
# Any combination of TCPIP, APPC, and NONE, separated by commas is valid.
# The setting specified by the DB2COMM environment variable is used to
# override which communication protocols are started, as implied by the
# tpname and svcename parameters in the database manager configuration
# file.
# If DB2COMM is undefined or set to null, the communications support
# implied by the definition of the service_name and/or tpname keywords
# in the database manager configuration file will determine which
# communications support will be enabled.
#-----------------------------------------------------------------------
DB2COMM=

#-----------------------------------------------------------------------
# DB2CHKPTR ·Default= null (OFF), any value (ON)“
# selectively turns pointer checking ON or OFF.
#-----------------------------------------------------------------------
DB2CHKPTR=OFF
```

Figure 58 (Part 2 of 2). Sample CICS Region Environment File

Chapter 10. Terminal Access to Your CICS/6000 Region

In scenario 1 all terminal handling is done by the mainframe. Now, in scenario 2, all terminal functions must be moved to AIX and CICS/6000.

In this chapter we examine the steps to be taken to fully implement our CICS region on the IBM RISC System/6000. The mainframe is no longer acting as the TOR, and the terminal handling functions must now be implemented on CICS/6000.

Considering that the mainframe is no longer available to us, we need to explore the new terminal options open to us:

- Replace all terminals with ASCII type devices, such as the IBM 3151 terminal, connected directly to the IBM RISC System/6000 or connected to a network using terminal servers or other IBM RISC System/6000s acting as terminal servers.
- Retain the current 3270 terminals. Connecting 3174s directly to a Token-Ring or Ethernet network using RPQ 8Q1041 enables existing 3270-type terminals to run tn3270 sessions and removes the requirement for such terminals to connect through a mainframe.

To enable the connection of a terminal to a CICS/6000 region, the terminal must be defined in the Terminal Definitions (WD) database file. This is analogous to an entry in the Terminal Control Table (TCT) on a mainframe CICS system.

Defining terminals permanently in the WD database would be inefficient and inflexible. CICS/6000 overcomes this difficulty by implementing a terminal autoinstall user exit. When a terminal requests a CICS/6000 session, an entry in in the WD database is created automatically. Using the TERM environment variable from the AIX environment, CICS/6000 selects the correct terminal model from the list in the WD database and uses it to create the new WD entry. If a match is not found for the TERM variable, CICS/6000 rejects the session request. The list of terminal models can be customized as needed.

The WD is deleted once the CICS/6000 session has terminated; CICS/6000 keeps only the WD of active sessions.

CICS/6000 supports:

- ASCII 3270 data stream
- 3270 terminal emulator
- 3270 telnet access—AIX, MVS, VM, OS/2 and other TCP/IP implementations.

CICS/6000 provides two client terminal emulations: cicsterm amd cicsteld. Both provide 3270 terminal emulation, but they do so in different ways.

10.1 cicsterm

This client program emulates a 3270 terminal with a CICS session. It provides screen management using curses primitives and communicates with the CICS/6000 region using 3270 data flows across Remote Procedure Calls (RPCs).

Because communications between the CICS/6000 region use RPCs, a system running cicsterm must be a member of the same DCE cell as the CICS/6000 region and must also have the Encina Executive installed. The cicsterm program is therefore limited to running on systems that can fulfill these requirements.

To connect to a CICS/6000 region using cicsterm, you must first log in as an AIX user, then authenticate yourself as a valid DCE user through the dce_login process. Cicsterm performs an implicit logon to CICS/6000 based on your DCE principal. CICS/6000 tries to match your DCE principal to one of its own user definitions. If it finds a match, you gain the security level of that CICS user. If no match is found, you can only access those CICS/6000 resources defined for public access. You can obtain further access rights by performing the CICS signon transaction, CESN. We strongly recommend that you have a CICS userid that matches your DCE principal userid.

The command to invoke cicsterm is shown in Figure 59.

```
cicsterm -r <cicsregion>
```

Figure 59. Invoking cicsterm

Specifying the CICS region `-r <cicsregion>` is optional. If not specified, the Cell Directory Service (CDS) namespace will be searched, and a list of all available CICS/6000 regions will be displayed.

10.2 cicsteld

The cicsteld process acts as an interface between CICS/6000 and a telnet client. Only telnet clients capable of emulating IBM 3270 terminals can be used to communicate with cicsteld. Fortunately such clients are available in almost all TCP/IP implementations.

Cicsteld can be invoked from the command line, but the more common approach is to register the cicsteld daemon with inetd as a subserver. A TCP/IP subserver is created to start a cicsteld process when a telnet call is made to the specified port on the server machine. The 3270 telnet client communicates with the cicsteld using ASCII 3270 data streams over TCP/IP, and cicsteld in turn communicates with the CICS/6000 region using RPCs in the same way as the cicsterm process. The system running cicsteld must therefore be installed in the same DCE cell as the CICS/6000 region.

When inetd starts a cicsteld process, cicsteld will service all incoming requests and begin communication with the specified CICS/6000 region. The same port can be used for all requests for connection to the region. A cicsteld process will be started for each 3270 telnet user.

10.2.1.1 Security

Cicsteld appears as a client application requesting services to the CICS/6000 region and must be authenticated by DCE before it can access the CICS/6000 region. This implies that a DCE principal must always be specified when initiating a cicsteld session. Because cicsteld is usually started by inetd when an incoming telnet client connection request arrives, you must also inform inetd which DCE principal it is to use when it starts cicsteld. This is done during cicsteld configuration, which we look at in 10.3, "Configuration of cicsteld."

When you issue the telnet connection request you are given the CICS transaction authority of the CICS userid that matches the DCE principal specified in the cicsteld command. If this userid has not been defined to CICS/6000, your transaction authority will be public and you will need to run CESN to gain higher authority. CESN can be specified as the initial transaction for the cicsteld command. In this case, CESN is invoked automatically when you sign on, forcing you to sign on to CICS/6000.

10.3 Configuration of cicsteld

Setting up cicsteld so that it is invoked automatically by inetd requires the following tasks:

1. Registering a TCP/IP service name and port
2. Associating a program with the TCP/IP service name
3. Storing a password in the DCE keytab file.

You need root permission to perform these tasks.

10.3.1.1 Registering a TCP/IP Service Name and Port

The command shown in Figure 60 establishes the port to which the 3270 telnet clients issue their request when trying to connect to a CICS/6000 region.

```
inetserv -a -S -v cicsteld -p tcp -n 5001
```

Figure 60. Establishing the 3270 Telnet Port

Our service name in this example is `cicsteld` and our port number is `5001`. This information modifies the /etc/services file, which tells inetd to invoke the specified service when a request comes in on the port. Check that the port number you use has not already been used in the /etc/services file.

10.3.1.2 Associating a Program with the TCP/IP Service Name

Once you have defined the service name, configure the inetd entry for that service by using the command shown in Figure 61.

```
inetserv -a -I -v cicsteld -p tcp -t stream -w nowait -U root \
-r /usr/bin/cicsteld.sh -R "cicsteld"
```

Figure 61. Configuring Inetd

This command modifies the /etc/inetd.conf file and indicates what should happen when a specific service is requested (in our case, cicsteld).

Issue the command shown in Figure 62 on page 68 to force inetd to reread the /etc/services and /etc/inetd.conf files, thereby making the changes effective.

```
refresh -s inetd
```

Figure 62. Refreshing Inetd

The DCE default cache file has AIX permissions that make it readable only by the AIX root user. You must therefore define that cicsteld has to be started by root.

Rather than directly specifying cicsteld as the program to be run, create an AIX shell script containing the cicsteld command and its parameters. We create the /usr/bin/cicsteld.sh file and add the lines as shown in Figure 63.

```
#!/bin/ksh
exec /usr/lpp/cics/v1.1/bin/cicsteld -p cterm -r SJA2109I -t CESN
```

Figure 63. cicsteld.sh

This example tells inetd to start a cicsteld connection each time the service "cicsteld" is requested, that is, every time a TCP/IP request arrives on port 5001. The cicsteld instance should connect to the CICS/6000 region, SJA2109I, as CICS user cterm and call CESN.

Remember to make the script file executable as shown in Figure 64.

```
chmod 755 /usr/bin/cicsteld.sh
```

Figure 64. Setting Executable Permissions for cicsteld.sh

10.3.1.3 Storing a Password in the DCE Keytab File

You should now define the CICS/6000 user specified in the cicsteld.sh script (cterm in our example). This user is the DCE principal whose credentials will be used to access the DCE CDS. The credentials must be made available in a keytab file on the IBM RISC System/6000 running the cicsteld process by using the rgy_edit command as shown in Figure 65.

```
# rgy_edit
rgy_edit=> ktadd -p cterm -pw <password_of_cterm>
rgy_edit=> quit
```

Figure 65. Setting the DCE Credentials for cicsteld.sh

Telnet clients can now access your CICS/6000 region.

Chapter 11. Batch Jobs

The application we migrated during this residency is completely interactive and online, with no batch job requirements.

We include this chapter for those users migrating applications from the mainframe CICS environment who need to duplicate the process of submitting jobs to batch, or for those users needing to start AIX scripts from within a CICS/6000 transaction.

Jobs in the AIX environment end up being converted to scripts. So just as we turned requests to run jobs over to JES* in the MVS environment, we need something to process script files for us in AIX. In AIX you can define a batch queue for this purpose.

To emulate the process of submitting jobs on CICS/6000, you will need to set up a batch queue that will process AIX script files and have a mechanism in CICS/6000 to deliver the jobs (scripts) to this batch queue.

11.1 Setting Up a Batch Queue

A batch queue allows you to make several requests to execute script files. This queue (of requests) enables the script files to be run asynchronously to the process that made the request.

To make a batch queue, add the lines shown in Figure 66 to your /etc/qconfig file:

```
batch:
        discipline = fcfs
        device = batchdev
batchdev:
        backend = /usr/bin/ksh
```

Figure 66. Adding a Batch Queue to /etc/qconfig

In the Figure 66, batch is the name of the queue, the device is a pointer to the next entry which indicates the program that is to process any entry that is placed on this queue. ksh is the name of the Korn shell, so your scripts that are placed on this queue will be processed by the Korn shell. The order in which they are processed is determined by the discipline. fcfs indicates that the scripts will be processed on a first come first serve basis.

Note that you must stopsrc -s qdaemon and startsrc -s qdaemon to pick up this new definition.

Once you have set up this queue you can lp -d batch yourscript to have the qdaemon run yourscript for you (lp is an AIX command to deliver something to a queue, -d indicates the name of the queue that follows, and yourscript is the name of the script you want executed).

The batch queue defined above will process scripts serially, but you may add multiple devices that point to multiple ksh backends. The qdaemon will redirect the standard input, standard output, and standard error of the executed script to /dev/null, but you may redirect the output of the commands in the script to a file if you want a history trail.

11.2 Getting Script or Job Request from CICS/6000 to Your Batch Queue

You can deliver a script request to the batch queue from CICS/6000 as follows:

1. Change the application code to construct a script command instead of constructing JCL.
2. Write the script request to a transient data queue.
3. Define a transient data queue to receive the script command and trigger (start) a CICS/6000 transaction to process the script command.
4. Write the CICS/6000 transaction to take script commands from the transient data queue and deliver them to a cicstermp command and define this transaction to CICS/6000.
5. Define a WD (workstation definition) to represent the cicstermp command.
6. Start a cicstermp command that will deliver the script request to the batch queue.

11.2.1 Change the Application Code to Produce Script Commands

The application code that constructs the JCL statements will have to be changed to construct script requests. If your code produced something like that shown in Figure 67, it would need to be changed to generate something like that shown in Figure 68. (Note that the script name is fully qualified.)

```
//CICSJOB  JOB (ACCT,INFO),"PROD CICS SUBMIT",CLASS=A
//PROCSTEP EXEC PROCXYZ,PARM="MONTHLY,REPORT5"
//EOJ
```

Figure 67. JCL Sample

```
/ibm/prodjobs/procxyz monthly report5
```

Figure 68. Script Sample

For a full explanation of JCL parameters and conversion, see Appendix F, “JCL Conversion” on page 137.

11.2.2 Write the Script Request to a Transient Data Queue

You may already be writing the records to a transient data queue. If you are using the SPOOLOPEN, SPOOLWRITE, and SPOOLCLOSE mechanism, you will have to change your program to write the script request to a transient data queue.

If you use the sample definitions in this chapter, you would write this script request record to the transient data queue named BTCH as shown in Figure 69 on page 71.

```
      EXEC CICS WRITEQ TD QUEUE("BTCH")
                       FROM(SCRIPT-REQUEST)
                       LENGTH(LENGTH OF SCRIPT-REQUEST)
                       END-EXEC.
```

Figure 69. Writing the Transient Data Queue

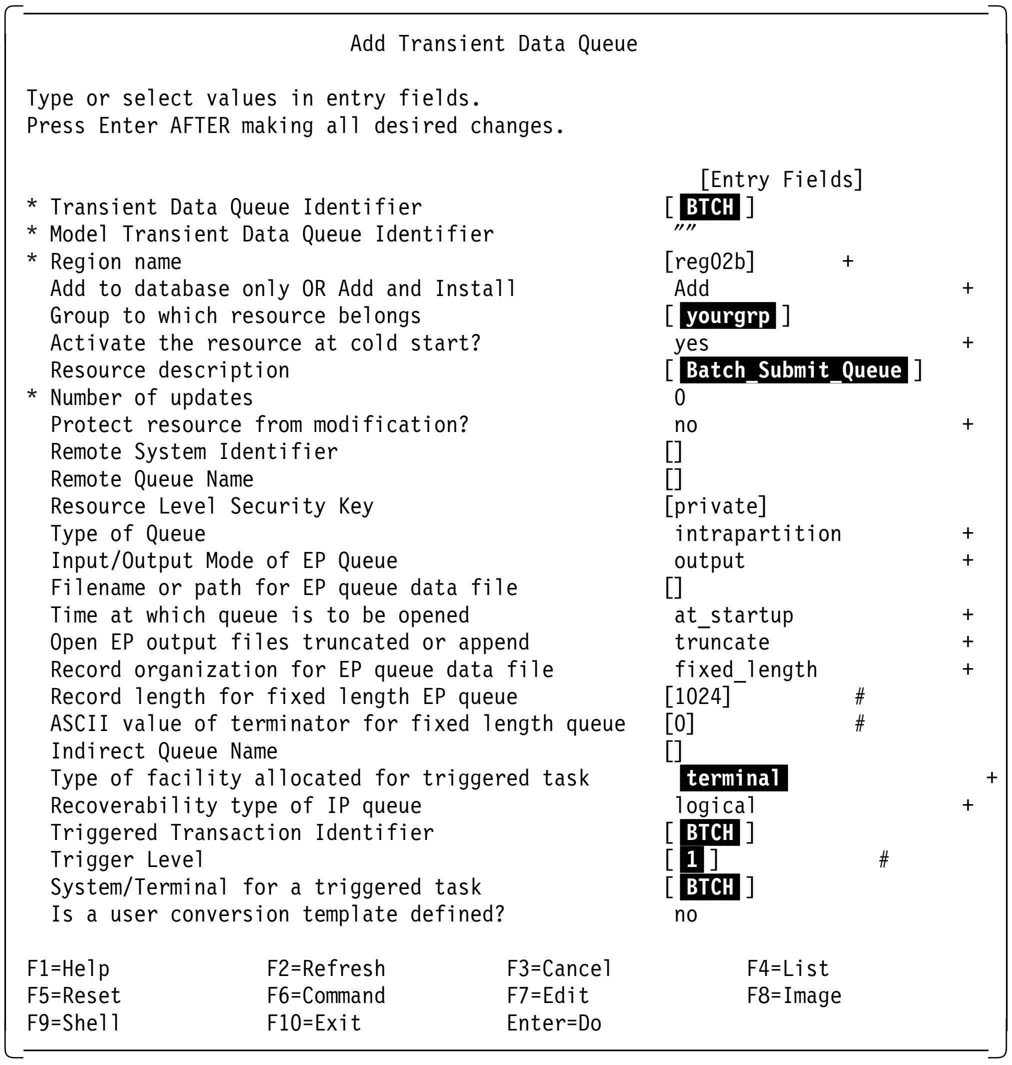
11.2.3 Define a Transient Data Queue for the Script Requests

Define a transient data queue as intrapartition with a trigger level of 1 to receive the script commands and start another CICS/6000 transaction to process the request. The triggering process is an optional feature of a transient data queue that allows it to start a transaction when items (in this case, one item) is placed on the transient data queue.

You can use the `cicsadd` command as shown in Figure 70, or `smitty cicsaddtdd` as indicated in Figure 71 on page 72.

```
$ cicsadd -c tdd -r <regions> BTCH GroupName=yourgrp \
> DestType=intrapartition FacilityType=Terminal \
> TriggeredTransId=BTCH TriggerLevel=1 FacilityId=BTCH
```

Figure 70. Defining a Transient Data Queue using cicsadd



```
                           Add Transient Data Queue

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                         [Entry Fields]
* Transient Data Queue Identifier                     [ BTCH ]
* Model Transient Data Queue Identifier                ""
* Region name                                         [reg02b]      +
  Add to database only OR Add and Install              Add                  +
  Group to which resource belongs                     [ yourgrp ]
  Activate the resource at cold start?                 yes                  +
  Resource description                                [ Batch_Submit_Queue ]
* Number of updates                                    0
  Protect resource from modification?                  no                   +
  Remote System Identifier                            []
  Remote Queue Name                                   []
  Resource Level Security Key                         [private]
  Type of Queue                                        intrapartition       +
  Input/Output Mode of EP Queue                        output               +
  Filename or path for EP queue data file             []
  Time at which queue is to be opened                  at_startup           +
  Open EP output files truncated or append             truncate             +
  Record organization for EP queue data file           fixed_length         +
  Record length for fixed length EP queue             [1024]         #
  ASCII value of terminator for fixed length queue    [0]            #
  Indirect Queue Name                                 []
  Type of facility allocated for triggered task         terminal             +
  Recoverability type of IP queue                      logical              +
  Triggered Transaction Identifier                    [ BTCH ]
  Trigger Level                                       [ 1 ]            #
  System/Terminal for a triggered task                [ BTCH ]
  Is a user conversion template defined?               no

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 71. Defining a Transient Data Queue Using SMIT

11.2.4 Write CICS/6000 Transaction to Process the Transient Data Queue

The transient data queue should trigger a simple transaction that reads the record from the intrapartition transient data queue on behalf of a terminal that you will define as a printer. The transaction would just SEND the script commands to the printer. Figure 72 shows a sample COBOL program to do this.

```
           IDENTIFICATION DIVISION.
           PROGRAM-ID.  BTCH.
           ENVIRONMENT DIVISION.
           DATA DIVISION.
           WORKING-STORAGE SECTION.
           01  RESP-FLD        PIC S9(8) COMP.
           01  QUELEN          PIC S9(4) COMP.
           01  CONTINUE-LOOP   PIC X(3) VALUE 'YES'.
           01  SCRIPT-REQUEST  PIC X(250).
           PROCEDURE DIVISION.
               PERFORM PRINT-QUEUE-ELEMENTS
                   THRU PRINT-QUEUE-ELEMENTS-EXIT
                      UNTIL CONTINUE-LOOP = 'NO'.
               EXEC CICS RETURN END-EXEC.
           PRINT-QUEUE-ELEMENTS.
               MOVE 250 TO QUELEN.
               EXEC CICS READQ TD QUEUE('BTCH')
                                  LENGTH(QUELEN)
                                  INTO(SCRIPT-REQUEST)
                                  RESP(RESP-FLD)
                                  END-EXEC.
               IF RESP-FLD = DFHRESP(QZERO) EXEC CICS RETURN END-EXEC.
         * MIGHT WANT TO CHECK FOR MORE ERRORS HERE.
               EXEC CICS SEND TEXT FROM(SCRIPT-REQUEST)
                                   PRINT
                                   END-EXEC.
           PRINT-QUEUE-ELEMENTS-EXIT.
               EXIT.
```

Figure 72. Sample COBOL Program for Writing to a Transient Data Queue

Start preparing this program by entering the source above into a file named btch.ccp. Compile the program using the command shown in Figure 73.

```
# cicstcl btch
```

Figure 73. Compiling btch.ccp

Then, copy the resulting executable to the region bin directory as shown in Figure 74.

```
#cp btch.gnt /var/cics_regions/<region>/bin
```

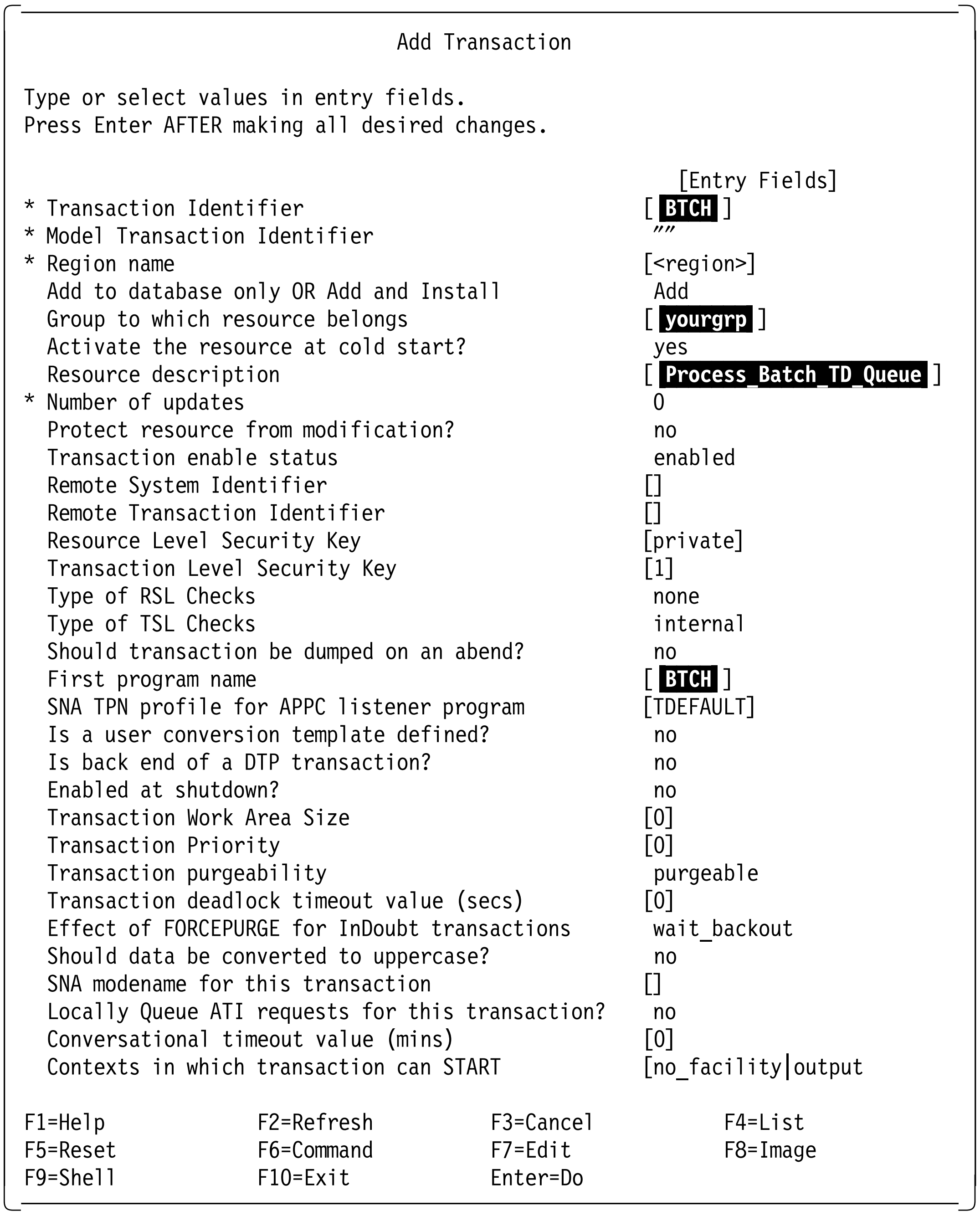
Figure 74. Copying btch.gnt

11.2.5 Define Your New Transaction to CICS/6000

You can define the transaction you wrote in the previous step to CICS/6000 with the `cicsadd` command as shown in Figure 75, or use `smitty cicsaddtd` and `smitty cicsaddpd` as shown in Figure 76 on page 75 and Figure 77 on page 76.

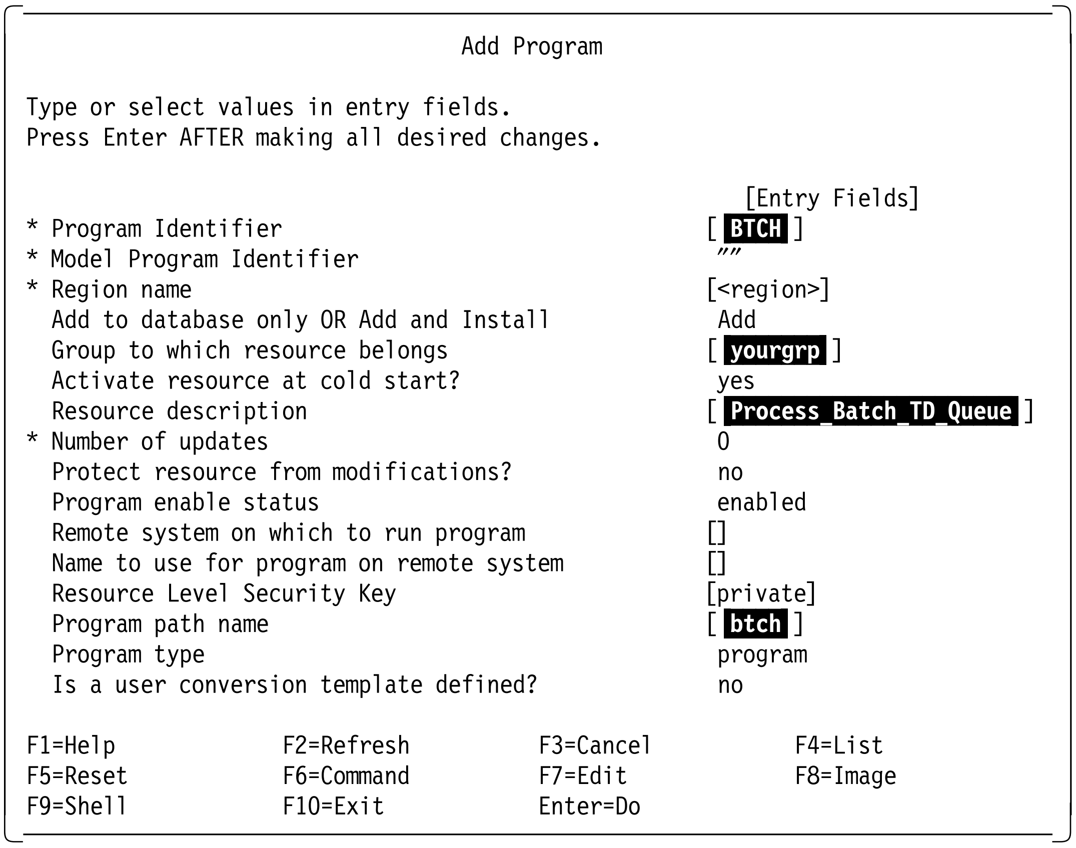
```
$ cicsadd -c td -r <regions> BTCH GroupName=yourgrp \
> ProgName=BTCH
$
$ cicsadd -c pd -r <regions> BTCH GroupName=yourgrp \
> PathName=btch
$
```

Figure 75. Defining the BTCH Transaction



```
                              Add Transaction

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                           [Entry Fields]
* Transaction Identifier                               [ BTCH ]
* Model Transaction Identifier                          ""
* Region name                                          [<region>]
  Add to database only OR Add and Install               Add
  Group to which resource belongs                      [ yourgrp ]
  Activate the resource at cold start?                  yes
  Resource description                                 [ Process_Batch_TD_Queue ]
* Number of updates                                     0
  Protect resource from modification?                   no
  Transaction enable status                             enabled
  Remote System Identifier                             []
  Remote Transaction Identifier                        []
  Resource Level Security Key                          [private]
  Transaction Level Security Key                       [1]
  Type of RSL Checks                                    none
  Type of TSL Checks                                    internal
  Should transaction be dumped on an abend?             no
  First program name                                   [ BTCH ]
  SNA TPN profile for APPC listener program            [TDEFAULT]
  Is a user conversion template defined?                no
  Is back end of a DTP transaction?                     no
  Enabled at shutdown?                                  no
  Transaction Work Area Size                           [0]
  Transaction Priority                                 [0]
  Transaction purgeability                              purgeable
  Transaction deadlock timeout value (secs)            [0]
  Effect of FORCEPURGE for InDoubt transactions         wait_backout
  Should data be converted to uppercase?                no
  SNA modename for this transaction                    []
  Locally Queue ATI requests for this transaction?      no
  Conversational timeout value (mins)                  [0]
  Contexts in which transaction can START              [no_facility|output

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 76. Defining the BTCH Transaction Using SMIT



```
                              Add Program

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                     [Entry Fields]
* Program Identifier                              [ BTCH ]
* Model Program Identifier                         ""
* Region name                                     [<region>]
  Add to database only OR Add and Install          Add
  Group to which resource belongs                 [ yourgrp ]
  Activate resource at cold start?                 yes
  Resource description                            [ Process_Batch_TD_Queue ]
* Number of updates                                0
  Protect resource from modifications?             no
  Program enable status                            enabled
  Remote system on which to run program           []
  Name to use for program on remote system        []
  Resource Level Security Key                     [private]
  Program path name                               [ btch ]
  Program type                                     program
  Is a user conversion template defined?           no

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

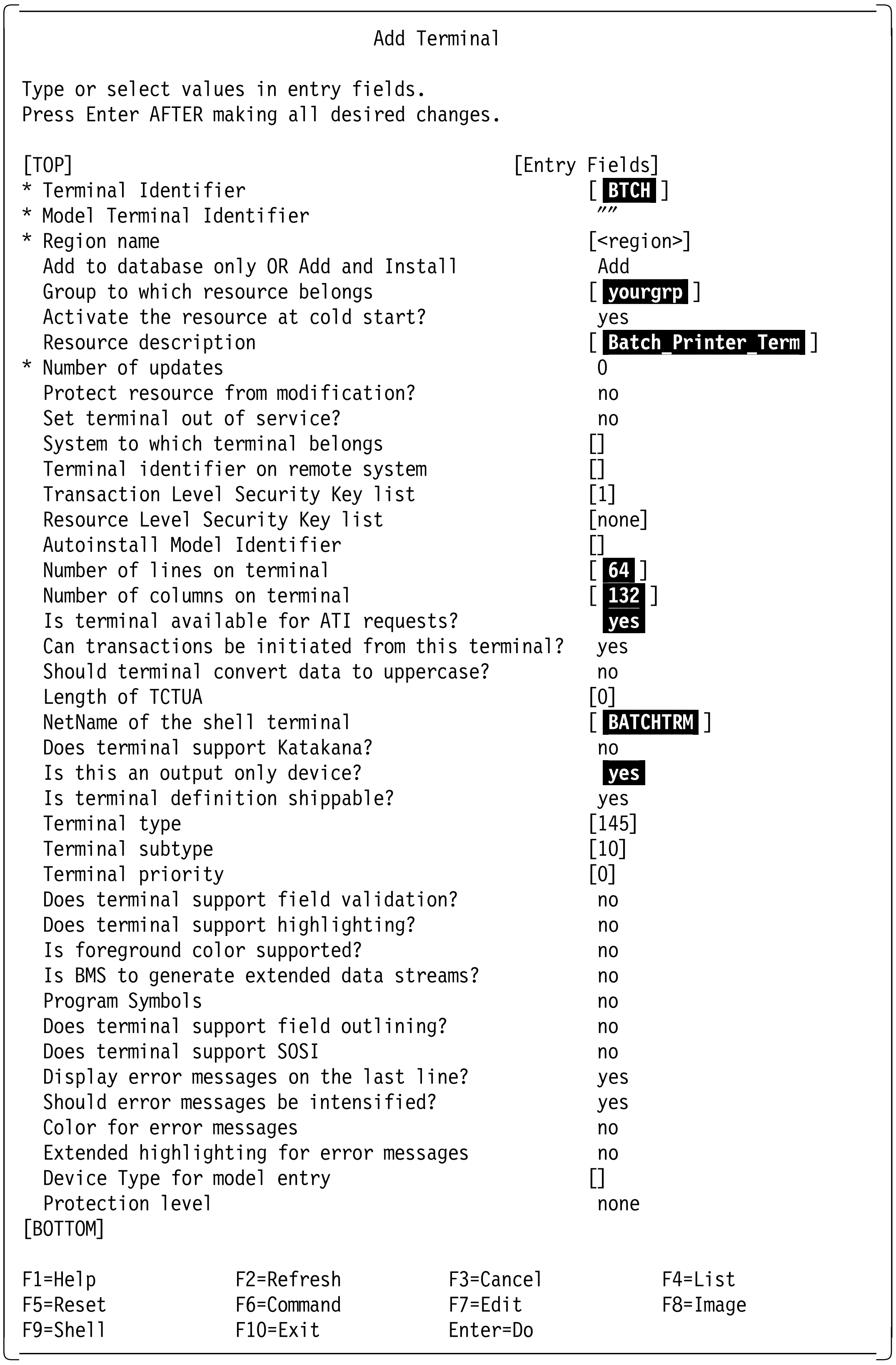
Figure 77. Defining the BTCH Program Using SMIT

11.2.6 Define a WD (Workstation Definitions)

You can define the printer terminal (WD) that is to receive the script command with the cicsadd command as shown in Figure 78, or use smitty cicsaddwd as shown in Figure 79 on page 77.

```
$ cicsadd -r <region> -c wd BTCH GroupName=yourgrp \
> CanStartATIs=yes NetName=BATCHTRM IsPrinter=yes \
> NumColumns=132 NumLines=64
```

Figure 78. Defining a Printer Terminal (WD)



```
                              Add Terminal

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                          [Entry Fields]
* Terminal Identifier                                  [BTCH]
* Model Terminal Identifier                             ""
* Region name                                          [<region>]
  Add to database only OR Add and Install               Add
  Group to which resource belongs                      [yourgrp]
  Activate the resource at cold start?                  yes
  Resource description                                 [Batch_Printer_Term]
* Number of updates                                     0
  Protect resource from modification?                   no
  Set terminal out of service?                          no
  System to which terminal belongs                     []
  Terminal identifier on remote system                 []
  Transaction Level Security Key list                  [1]
  Resource Level Security Key list                     [none]
  Autoinstall Model Identifier                         []
  Number of lines on terminal                          [64]
  Number of columns on terminal                        [132]
  Is terminal available for ATI requests?               yes
  Can transactions be initiated from this terminal?     yes
  Should terminal convert data to uppercase?            no
  Length of TCTUA                                      [0]
  NetName of the shell terminal                        [BATCHTRM]
  Does terminal support Katakana?                       no
  Is this an output only device?                        yes
  Is terminal definition shippable?                     yes
  Terminal type                                        [145]
  Terminal subtype                                     [10]
  Terminal priority                                    [0]
  Does terminal support field validation?               no
  Does terminal support highlighting?                   no
  Is foreground color supported?                        no
  Is BMS to generate extended data streams?             no
  Program Symbols                                       no
  Does terminal support field outlining?                no
  Does terminal support SOSI                            no
  Display error messages on the last line?              yes
  Should error messages be intensified?                 yes
  Color for error messages                              no
  Extended highlighting for error messages              no
  Device Type for model entry                          []
  Protection level                                      none
[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 79. Defining a Printer Terminal (WD) Using SMIT

11.2.7 Start the cicstermp Command

The `cicstermp` command is used to receive the script command and turn it over to the batch queue. The `cicstermp` command shown in Figure 80 would be started only after CICS/6000 has been started.

```
# dce_login <userid> <password>
# cicstermp -r <region> -n BATCHTRM -P ′lp -c -d batch′
```

Figure 80. Starting the cicstermp Command

11.3 Job Scheduling Systems

On a host system batch jobs are frequently part of a complex job stream with many dependencies on other jobs or job streams, time of day, and other events. In this section we present a brief overview of the available job scheduling systems for AIX.

11.3.1 IBM Job Scheduler for AIX

Although not available for testing at the time of writing this document, this new product could play a very important role in application environments with a heavy batch processing load.

IBM Job Scheduler for AIX (Job Scheduler) manages the complexity of scheduling, initiating, and monitoring the regular background workload in RISC System/6000 environments. Job Scheduler initiates work following the policies and rules your system administrators specify, and it monitors the workload through completion. It automatically detects jobs that have failed, and it can restart a failed job and optionally take a user-defined corrective action before restart. Job Scheduler produces daily and weekly workload execution plans. In addition, Job Scheduler′s job and execution logs provide an audit trail of execution.

11.3.2 OPC Tracker Agent for AIX

OPC/ESA* is IBM′s licensed program for planning and scheduling production workloads. OPC/ESA V1 R3 includes the OPC Tracker Agent for AIX, which, together with the corresponding host component, provides full function workload management from OPC/ESA.

The benefits are:

- High degree of automation
- Reduction of the complexity of enterprise workload management
- Real-time resource tracking and monitoring
- Limitation of contention for critical resources
- Planning of operations based on actual resource availability
- Avoidance of bottlenecks allowing for higher throughput.

11.3.3 AIX Cron Facility

The AIX cron daemon can run commands, for example, shell scripts, automatically. Regularly scheduled commands can be specified according to instructions contained in the crontab files. The crontab command is used to submit, edit, list, or remove commands to be run by the cron daemon at regularly scheduled intervals.

To add an entry to your crontab file enter the command as shown in Figure 81.

```
crontab -e
```

Figure 81. Editing Your crontab File

Figure 82 shows a sample crontab file. You can add entries for commands to be scheduled to this file. After you have saved your crontab file, the cron daemon is notified of the change, and your command is scheduled. Please refer to your *IBM AIX Version 3.2 for RISC System/6000 Commands Reference* manual for a detailed description of the syntax for crontab entries.

```
# @(#)08 1.15  com/cmd/cntl/cron/root, bos, bos320 9/9/91 06:04:47
#
# COMPONENT_NAME: (CMDCNTL) commands needed for basic system needs
#
# FUNCTIONS:
#
# ORIGINS: 27
#
# (C) COPYRIGHT International Business Machines Corp. 1989,1991
# All Rights Reserved
# Licensed Materials - Property of IBM
#
# US Government Users Restricted Rights - Use, duplication or
# disclosure restricted by GSA ADP Schedule Contract with IBM Corp.
#
#0 3 * * * /etc/skulker
#45 2 * * 0 /usr/lib/spell/compress
#45 23 * * * ulimit 5000; /usr/lib/smdemon.cleanu > /dev/null
0 11 * * * /usr/bin/errclear -d S,O 30
0 12 * * * /usr/bin/errclear -d H 90
01 4 * * * /etc/lpp/diagnostics/bin/test_batt 1>/dev/null 2>/dev/null
01 3 * * * /etc/lpp/diagnostics/bin/run_ela 1>/dev/null 2>/dev/null
0 * * * * echo The hour is date . >/dev/console
```

Figure 82. Sample crontab File

Chapter 12. Transferring VSAM Files

Encina SFS files on AIX are the equivalent of VSAM files on the MVS system. Applications that read VSAM files through the CICS API can run unchanged if the VSAM files are transferred and converted to SFS files. In this chapter we describes how to define an SFS file and how to load the VSAM data into it. The example presented in this chapter uses the CICS-supplied transaction, CALF, to load the VSAM data into your SFS file. This exercise requires an ISC link to be set up between your CICS MVS and your CICS/6000 system. In 12.6, “Transferring VSAM Data without an ISC Link” on page 87 we discuss transferring your VSAM data if you do not have an ISC link.

12.1 Creating an SFS Schema File

An SFS schema file contains the data for you to create files and indexes on an SFS server. An SFS schema file is an AIX stanza file. To create an SFS schema file enter the command shown in Figure 83.

```
smit cicsemptyscd
```

Figure 83. Creating an SFS Schema File

You will be prompted for the name of the schema file you want to create. This file can be in any location and does not belong to a particular SFS server.

12.2 Creating an SFS Schema

Once you have created the SFS schema file you can start creating your SFS schema. An SFS schema is stored within an SFS schema file. To work with SFS schema files enter the command shown in Figure 84.

```
smit cicsoperationscd
```

Figure 84. Working with SFS Schema Files

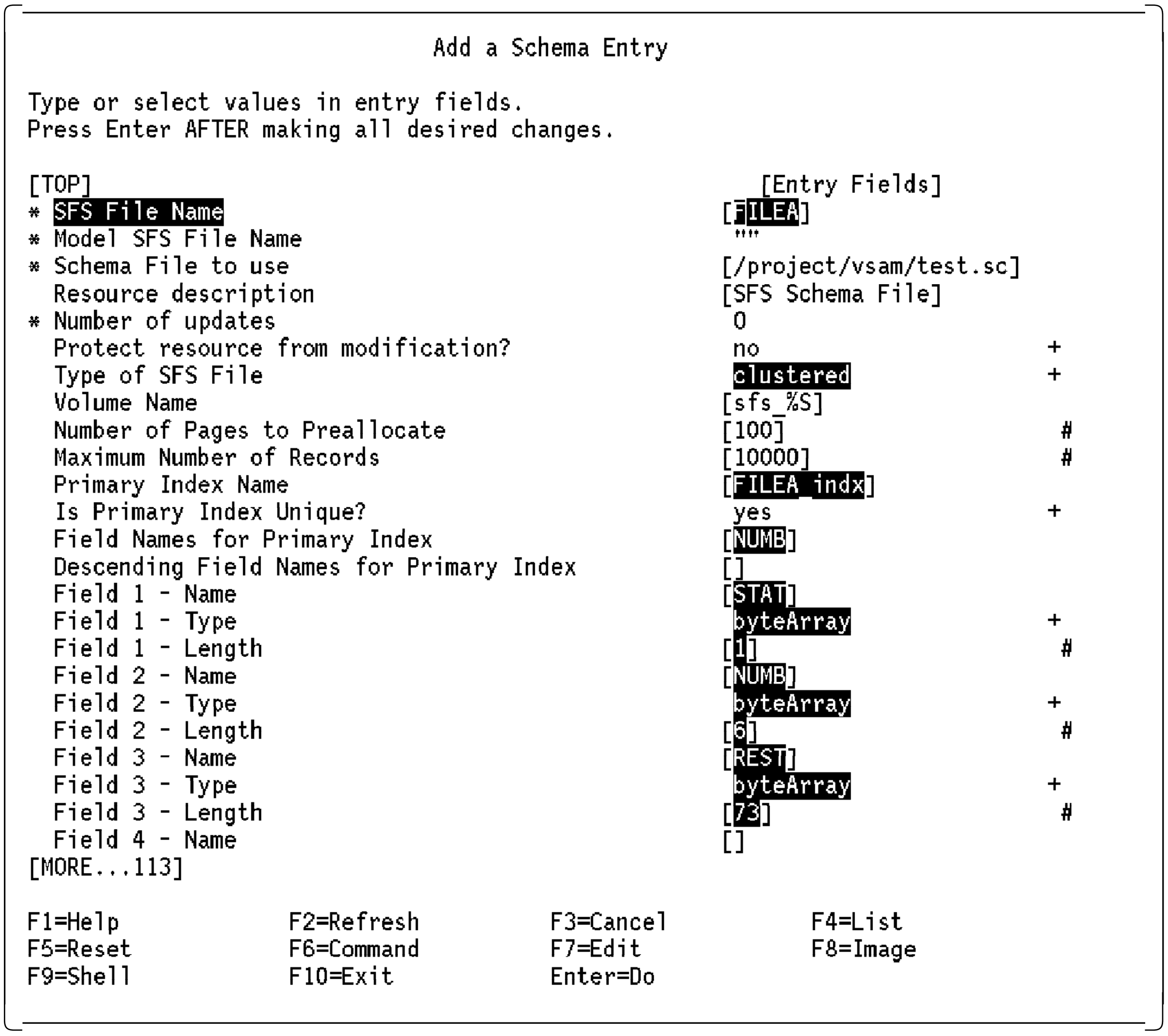
Select the *Change Working Schema File* option to select the SFS schema file you created. You can now select the *Add a new Schema* option and a new schema will be added to your schema file. The values that you have to enter for your schema depend on the definition of your VSAM file. You can choose any name for the SFS file name and for the primary index name; however, you have to refer to these names when you make the CICS file definitions for your CICS region. Table 4 compares the VSAM and SFS file types.

Table 4. VSAM and SFS File Types

VSAM Data Set Type	SFS File Type
Key-sequenced data set (KSDS)	Clustered files
Entry-sequenced data set (ESDS)	Entry-sequenced files
Relative record data set (RRDS)	Relative files

You have to choose the SFS file type that corresponds to your VSAM data set type.

You must now define your fields and indexes. Apart from the index fields you do not have to define each field separately. It suffices to define just one nonindex field with the correct length, presuming the nonindex fields are adjacent. For all fixed length fields you can use the data type bytearray. Refer to Chapter 9, “File Services,” in the *AIX CICS/6000 Application Programming Guide* for information on how to define variable length records and restrictions that apply. Figure 85 shows you an example of an SFS clustered file schema.



```
                              Add a Schema Entry

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* SFS File Name                                      [FILEA]
* Model SFS File Name                                 ""
* Schema File to use                                 [/project/vsam/test.sc]
  Resource description                               [SFS Schema File]
* Number of updates                                   0
  Protect resource from modification?                 no                    +
  Type of SFS File                                    clustered             +
  Volume Name                                        [sfs_%S]
  Number of Pages to Preallocate                     [100]                   #
  Maximum Number of Records                          [10000]                 #
  Primary Index Name                                 [FILEA_indx]
  Is Primary Index Unique?                            yes                   +
  Field Names for Primary Index                      [NUMB]
  Descending Field Names for Primary Index           []
  Field 1 - Name                                     [STAT]
  Field 1 - Type                                      byteArray             +
  Field 1 - Length                                   [1]                     #
  Field 2 - Name                                     [NUMB]
  Field 2 - Type                                      byteArray             +
  Field 2 - Length                                   [6]                     #
  Field 3 - Name                                     [REST]
  Field 3 - Type                                      byteArray             +
  Field 3 - Length                                   [73]                    #
  Field 4 - Name                                     []
[MORE...113]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 85. SFS Clustered File Schema

12.3 Applying an SFS Schema to an Encina SFS Server

To create the Encina SFS file you select the *Apply Schemas to an Encina SFS Server (with all Indexes)* option on the *Manage Schemas for Encina SFS Servers* smitty screen. Select the Encina SFS files that you want to apply to your Encina SFS server. On the next screen you can enter the Encina SFS server to which you want to apply the schemas.

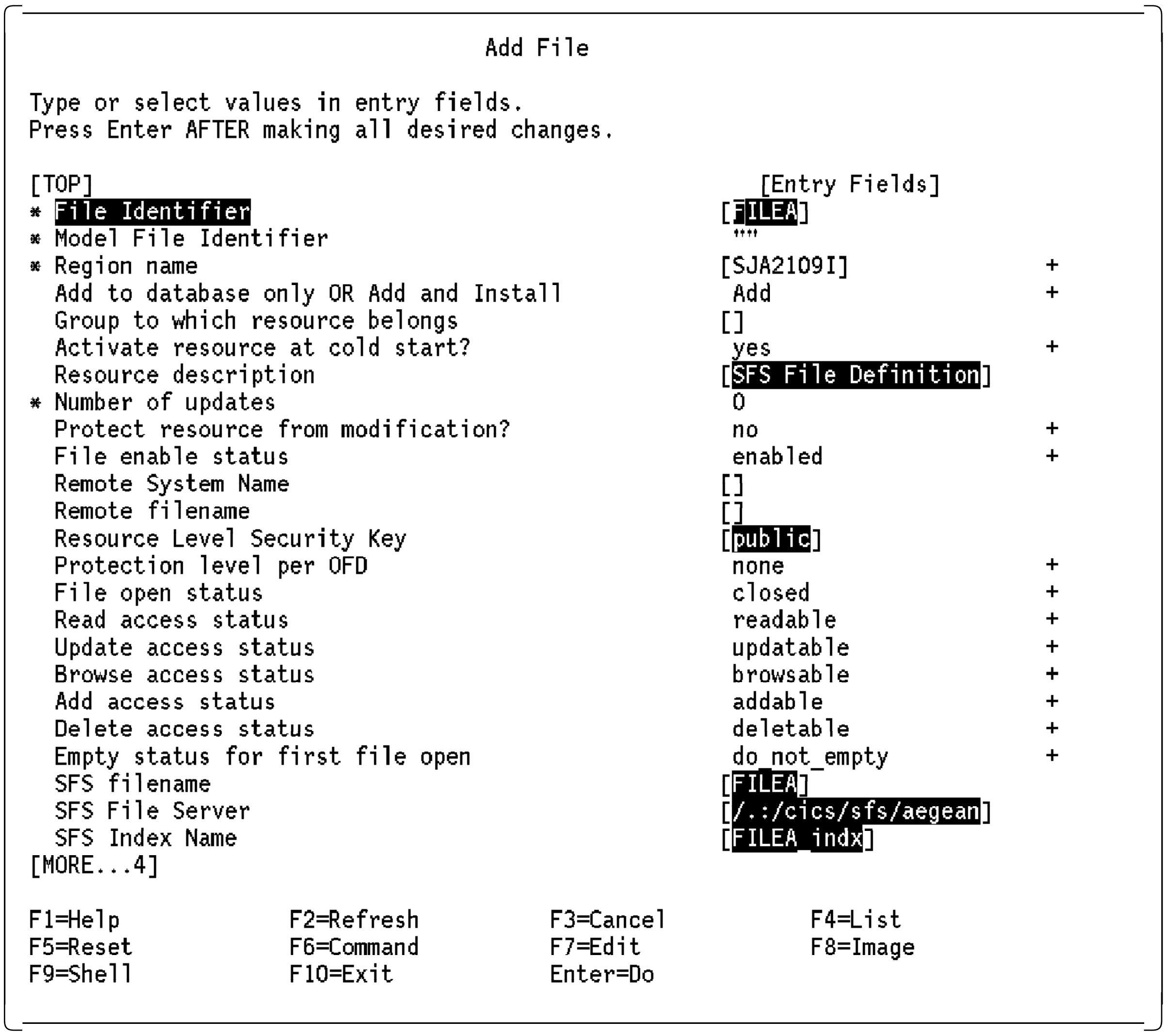
12.4 Adding a File Definition to Your CICS Region

In order to use your SFS file from within CICS you must add a CICS file definition for it. Enter the command shown in Figure 86.

```
smit cicsaddfd
```

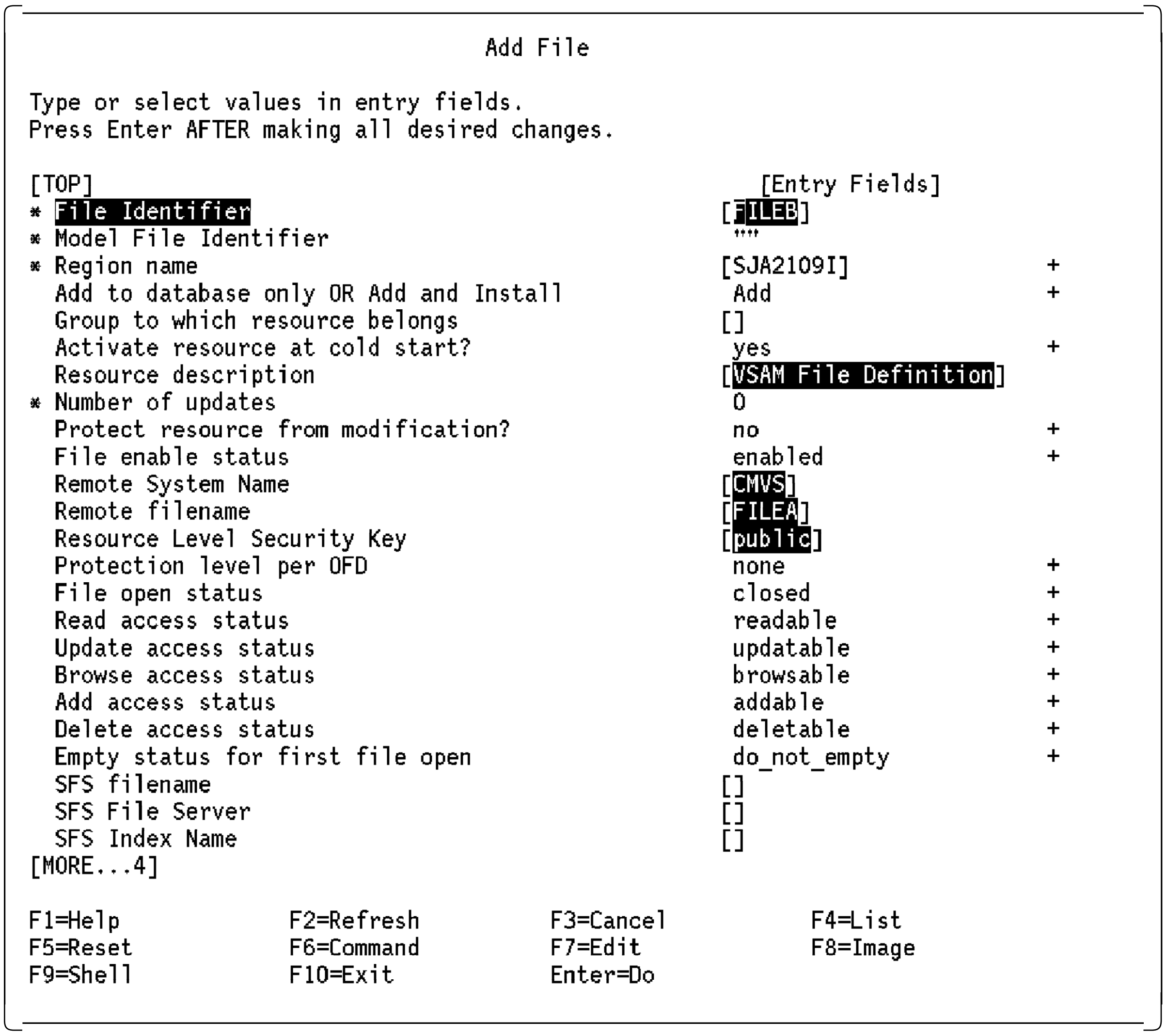
Figure 86. Adding a File Definition to Your CICS Region

Specify the SFS file server, SFS file name, and SFS primary index name. Specify *empty* for the empty status for the first file open. Remember to set the correct security attributes. You now have an empty local SFS file accessible by CICS into which you can load your VSAM data. If you want to use the CICS-supplied CALF transaction to load your VSAM data into your SFS file, you must now define a remote file, pointing to the VSAM file on your CICS MVS system. Proceed as for adding the local file definition to your CICS region, but enter the values for your remote system and the file name of the VSAM file on the remote system. You do not need to enter any values for the SFS file server, file name, or primary index name. Figure 87 on page 84 shows you an example of a definition for a local file, and Figure 88 on page 85 shows you an example of a definition for a remote file.



```
                          Add File

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* File Identifier                                   [FILEA]
* Model File Identifier                              ""
* Region name                                       [SJA2109I]                +
  Add to database only OR Add and Install            Add                      +
  Group to which resource belongs                   []
  Activate resource at cold start?                   yes                      +
  Resource description                              [SFS File Definition]
* Number of updates                                  0
  Protect resource from modification?                no                       +
  File enable status                                 enabled                  +
  Remote System Name                                []
  Remote filename                                   []
  Resource Level Security Key                       [public]
  Protection level per OFD                           none                     +
  File open status                                   closed                   +
  Read access status                                 readable                 +
  Update access status                               updatable                +
  Browse access status                               browsable                +
  Add access status                                  addable                  +
  Delete access status                               deletable                +
  Empty status for first file open                   do_not_empty             +
  SFS filename                                      [FILEA]
  SFS File Server                                   [/.:/cics/sfs/aegean]
  SFS Index Name                                    [FILEA_indx]
[MORE...4]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 87. File Definition for Local File

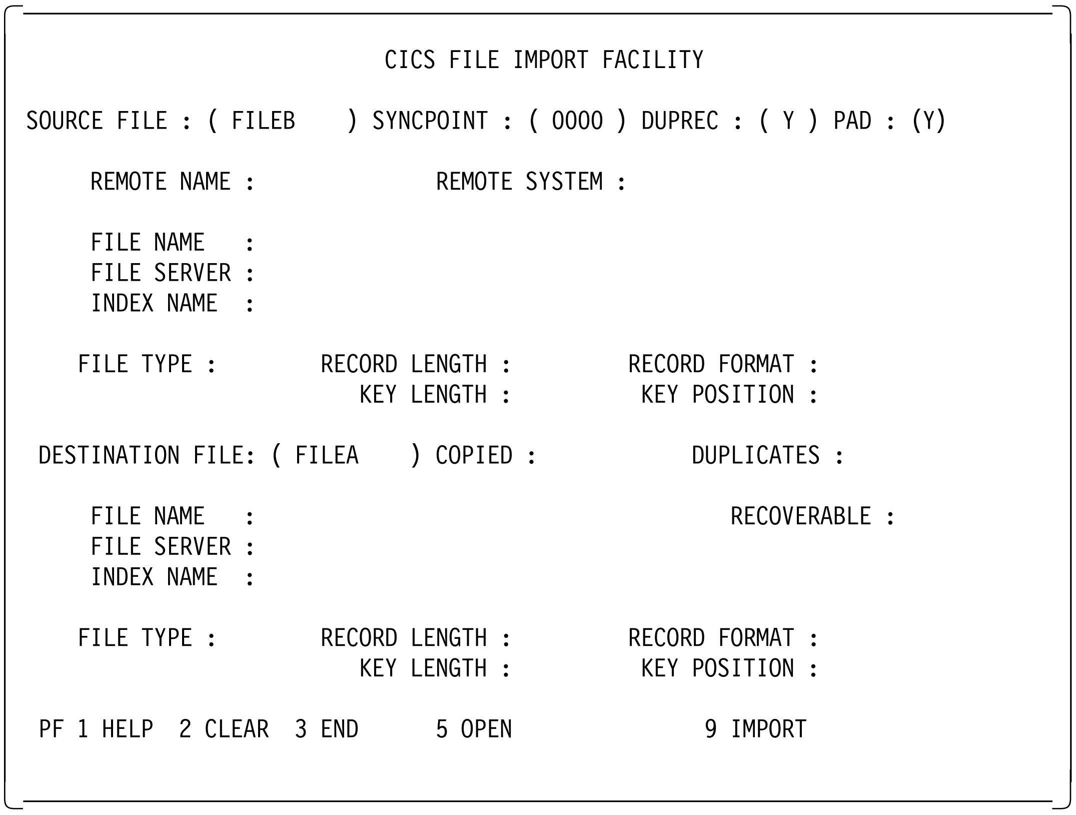


```
                              Add File

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* File Identifier                                    [FILEB]
* Model File Identifier                               ""
* Region name                                        [SJA2109I]             +
  Add to database only OR Add and Install             Add                   +
  Group to which resource belongs                    []
  Activate resource at cold start?                    yes                   +
  Resource description                               [VSAM File Definition]
* Number of updates                                   0
  Protect resource from modification?                 no                    +
  File enable status                                  enabled               +
  Remote System Name                                 [CMVS]
  Remote filename                                    [FILEA]
  Resource Level Security Key                        [public]
  Protection level per OFD                            none                  +
  File open status                                    closed                +
  Read access status                                  readable              +
  Update access status                                updatable             +
  Browse access status                                browsable             +
  Add access status                                   addable               +
  Delete access status                                deletable             +
  Empty status for first file open                    do_not_empty          +
  SFS filename                                       []
  SFS File Server                                    []
  SFS Index Name                                     []
[MORE...4]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 88. File Definition for Remote File

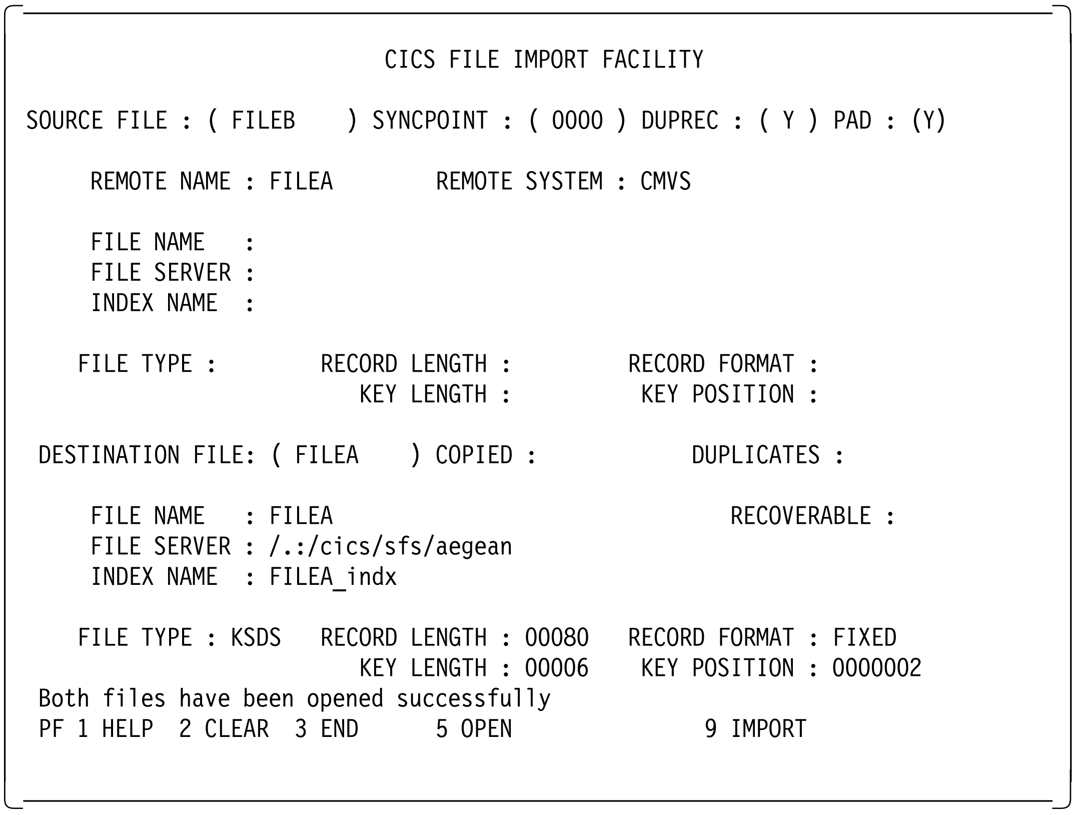
12.5 Loading the VSAM Data into Your SFS File

You can use the CICS-supplied CALF transaction to load the data from your VSAM file into your SFS file. At your CICS terminal enter the transaction name CALF. On the next screen enter the name of the source file (which is the file you defined as remote) and the destination file (which is your SFS file to be loaded) (see Figure 89 on page 86). You can now press function key 5 to open both of these files. If you get the message “Both files have been opened successfully,” you can press function key 9 to import the data (see Figure 90 on page 86). After a short time, you should receive the message “Copy has completed successfully - ′nn′ record(s) copied,” where nn refers to the number of records copied. You can now run your CICS/6000 transactions and access your SFS file instead of the VSAM file. Please refer to Chapter 2, “Data Conversion (CALF)” in *AIX CICS/6000 CICS-supplied Transactions* for more information on the CALF transaction.



```
                            CICS FILE IMPORT FACILITY

SOURCE FILE : ( FILEB    ) SYNCPOINT : ( 0000 ) DUPREC : ( Y ) PAD : (Y)

     REMOTE NAME :             REMOTE SYSTEM :

     FILE NAME   :
     FILE SERVER :
     INDEX NAME  :

    FILE TYPE :       RECORD LENGTH :         RECORD FORMAT :
                         KEY LENGTH :          KEY POSITION :

 DESTINATION FILE: ( FILEA    ) COPIED :           DUPLICATES :

     FILE NAME   :                                    RECOVERABLE :
     FILE SERVER :
     INDEX NAME  :

    FILE TYPE :       RECORD LENGTH :         RECORD FORMAT :
                         KEY LENGTH :          KEY POSITION :

 PF 1 HELP  2 CLEAR  3 END      5 OPEN              9 IMPORT
```

Figure 89. First Screen of CALF Transaction



```
                            CICS FILE IMPORT FACILITY

SOURCE FILE : ( FILEB    ) SYNCPOINT : ( 0000 ) DUPREC : ( Y ) PAD : (Y)

     REMOTE NAME : FILEA       REMOTE SYSTEM : CMVS

     FILE NAME   :
     FILE SERVER :
     INDEX NAME  :

    FILE TYPE :       RECORD LENGTH :         RECORD FORMAT :
                         KEY LENGTH :          KEY POSITION :

 DESTINATION FILE: ( FILEA    ) COPIED :           DUPLICATES :

     FILE NAME   : FILEA                              RECOVERABLE :
     FILE SERVER : /.:/cics/sfs/aegean
     INDEX NAME  : FILEA_indx

    FILE TYPE : KSDS   RECORD LENGTH : 00080   RECORD FORMAT : FIXED
                         KEY LENGTH : 00006    KEY POSITION : 0000002
 Both files have been opened successfully
 PF 1 HELP  2 CLEAR  3 END      5 OPEN              9 IMPORT
```

Figure 90. Second Screen of CALF Transaction

12.6 Transferring VSAM Data without an ISC Link

If you cannot use the CICS ISC functions to transfer your VSAM data to your SFS file system, you will have to copy your VSAM data to a sequential file using the REPRO command (see Figure 91).

```
//REPRO    JOB (999,POK),'REPRO JOB         ',
//             CLASS=A,MSGCLASS=T,MSGLEVEL=(1,1)
//*
//REPRO   EXEC PGM=IDCAMS
//SYSPRINT  DD SYSOUT=*
//DDIN      DD DSN=MY.VSAM.DATASET,DISP=SHR
//DDOUT     DD DISP=(NEW,CATLG,DELETE),
//             SPACE=(CYL,(5,1)),UNIT=SYSDA,VOL=SER=STCIC1,
//             DCB=(LRECL=80,RECFM=FB,BLKSIZE=8000),
//             DSN=MY.SEQUENTL.FILE
//SYSIN     DD *
 REPRO INFILE(DDIN) OUTFILE(DDOUT)
/*
//*
```

Figure 91. Flattening a VSAM File Using REPRO

You must then transfer the sequential file to your AIX system (see Chapter 5, "Transferring Sequential Files" on page 15). If your VSAM file contains both text and binary or decimal fields, you have to do the EBCDIC to ASCII conversion on the sequential file manually and transfer it using the binary mode. In Appendix G, "EBCDIC to ASCII Conversion" on page 143 you can find a sample program to convert only text fields in a file from EBCDIC to ASCII. You now have to write a program to load the data from the converted and transferred sequential file into your SFS file system.

Chapter 13. Printing

This chapter refers to different ways of dealing with printing in migration scenarios 1 and 2.

13.1 Scenario 1

To generate printed output in a CICS/MVS system, you have the following options:

- Hardware print key feature
- CICS print key feature
- ISSUE PRINT command
- Print transaction started on a printer terminal
- Transient data queue with a trigger level
- Writing to the JES spool.

We discuss below the options for scenario 1.

13.1.1 Hardware Print Key Feature

If you configure your terminal control unit to support the key print feature, you can print the contents of your display screen to the hardware-designated printer, by pressing the PrintScreen key. This process involves only the display terminal, the control unit, and the printer attached to it. Neither the host processor nor CICS nor your application program can control this process. So, this option remains valid for scenario 1.

13.1.2 CICS Print Key Feature and ISSUE PRINT Command

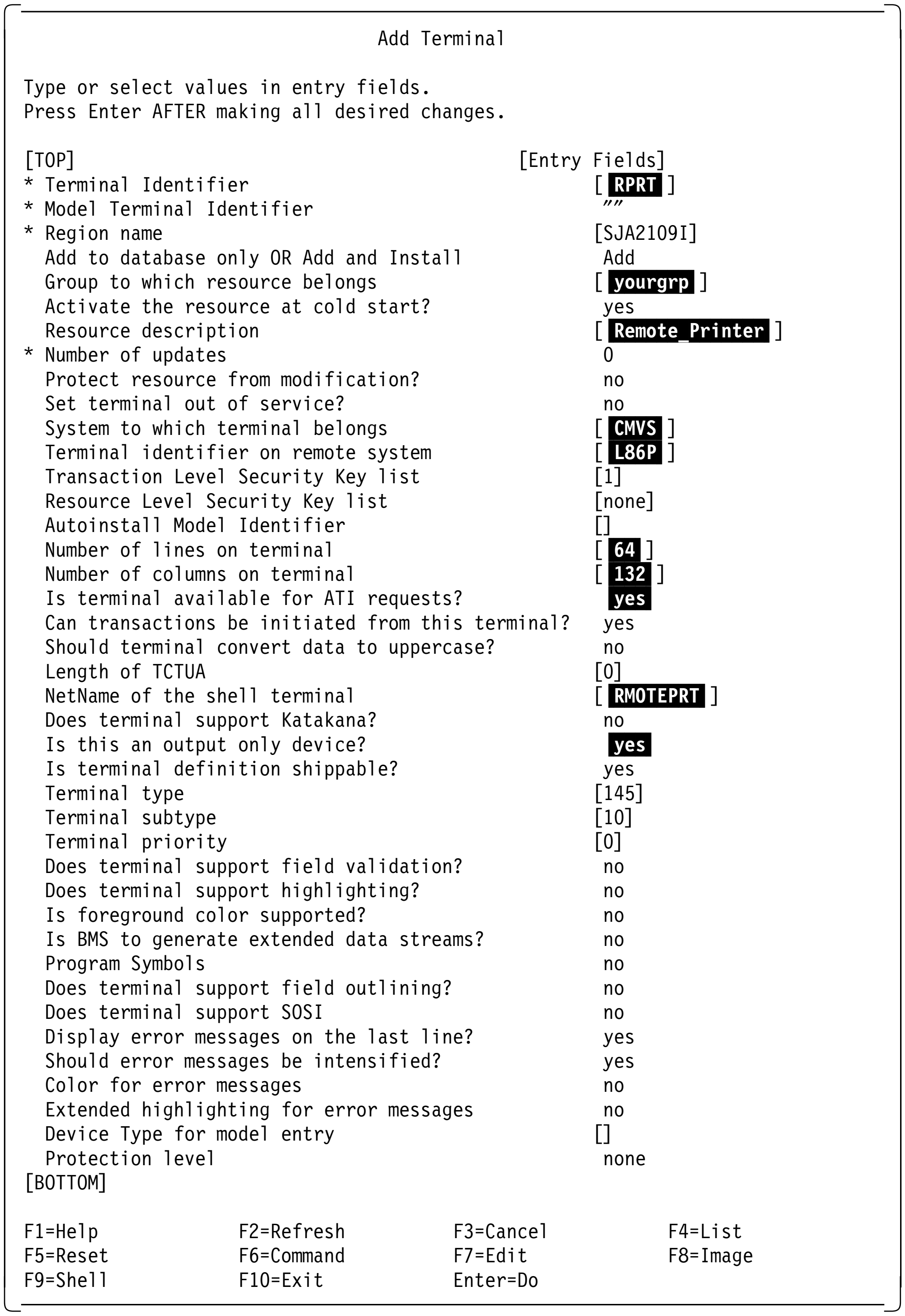
You can reserve a special key within CICS to allow your users to print the screen contents using a program access (PA) key or embed an ISSUE PRINT command in your program. The screen copy produced by these means goes to the printer assigned in the PRINTTO parameter of the TERMINAL definition of your display terminal.

Even if you are connected by means of the CRTE transaction to the remote CICS/6000 system or using a transaction defined as remote in the host CICS/ESA, the PRINTTO mechanism is still active in the real host terminal, so when you press the PA key or the remote program issues the ISSUE PRINT request, you get the screen printed on the printer assigned to the terminal. Therefore, these printing options are valid for scenario 1.

13.1.3 Print Transaction Started on a Printer

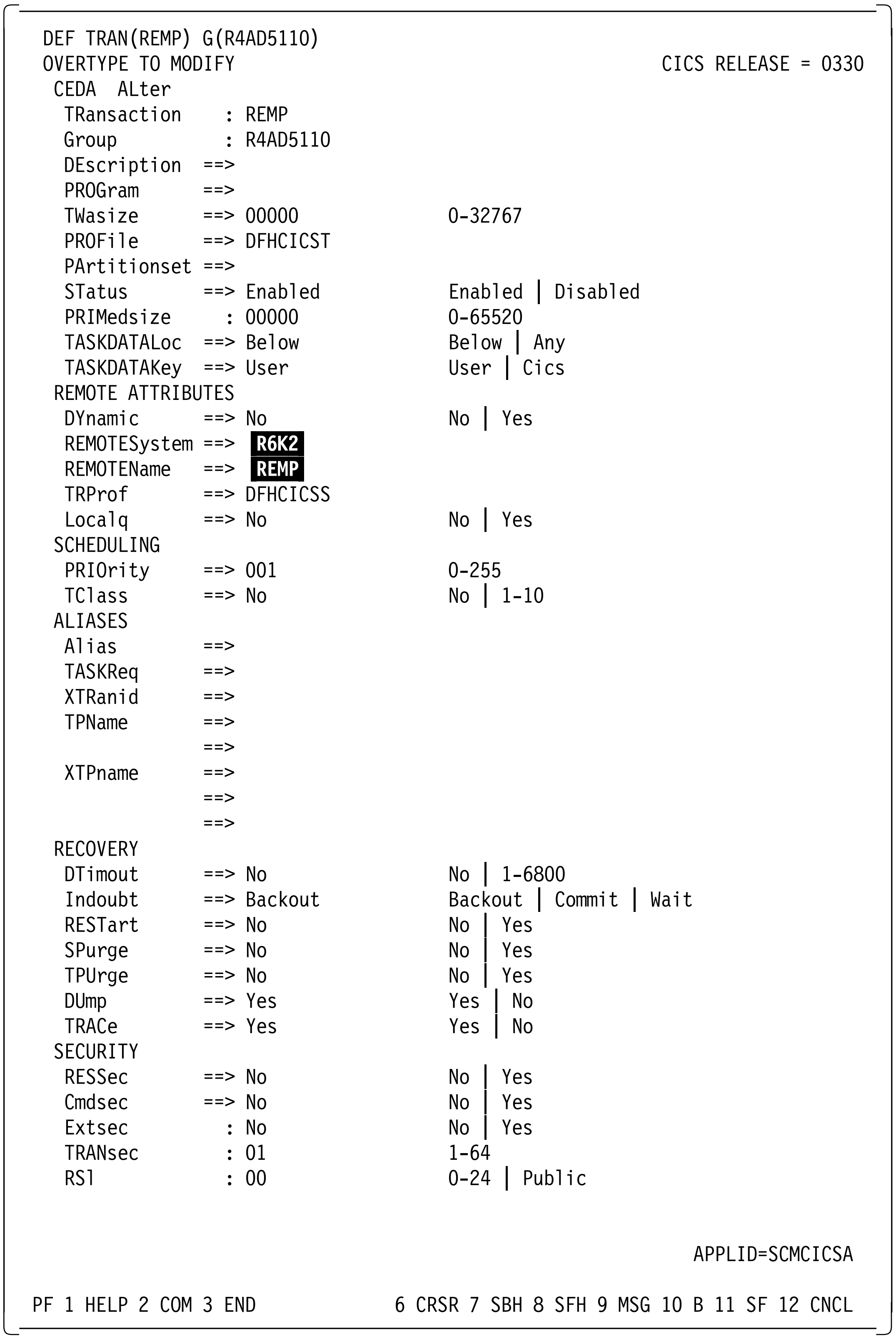
You can use the START command to initiate a secondary CICS task on a printer by referring the printer name in the TERMID parameter and passing the data you want to print in the FROM area. This facility is called automatic transaction initiation (ATI). The print task conveniently formats and arranges the data received and prints it using BMS SEND commands with the PRINT option.

To use ATI in scenario 1, you must define the printer terminal as remote in AIX (see in Figure 92 on page 90), define the transaction that is ATI-initiated as remote in MVS (see Figure 93 on page 91), and set the SNA TPN profile for the transaction in AIX (see Figure 94 on page 92). These actions will enable the host system to identify the transaction.



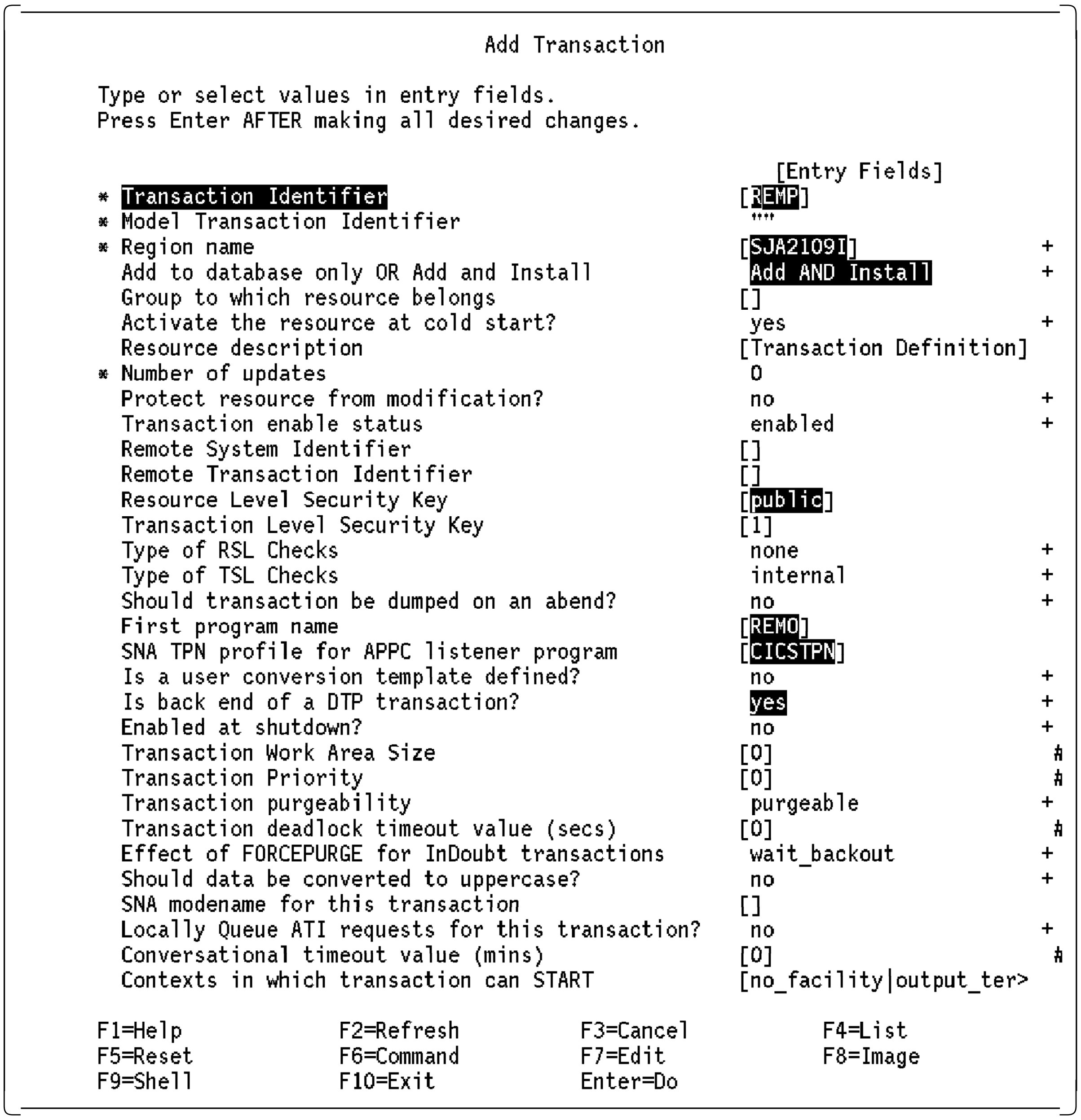
```
                              Add Terminal

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                              [Entry Fields]
* Terminal Identifier                                  [ RPRT ]
* Model Terminal Identifier                             ""
* Region name                                          [SJA2109I]
  Add to database only OR Add and Install               Add
  Group to which resource belongs                      [ yourgrp ]
  Activate the resource at cold start?                  yes
  Resource description                                 [ Remote_Printer ]
* Number of updates                                     0
  Protect resource from modification?                   no
  Set terminal out of service?                          no
  System to which terminal belongs                     [ CMVS ]
  Terminal identifier on remote system                 [ L86P ]
  Transaction Level Security Key list                  [1]
  Resource Level Security Key list                     [none]
  Autoinstall Model Identifier                         []
  Number of lines on terminal                          [ 64 ]
  Number of columns on terminal                        [ 132 ]
  Is terminal available for ATI requests?               yes
  Can transactions be initiated from this terminal?     yes
  Should terminal convert data to uppercase?            no
  Length of TCTUA                                      [0]
  NetName of the shell terminal                        [ RMOTEPRT ]
  Does terminal support Katakana?                       no
  Is this an output only device?                        yes
  Is terminal definition shippable?                     yes
  Terminal type                                        [145]
  Terminal subtype                                     [10]
  Terminal priority                                    [0]
  Does terminal support field validation?               no
  Does terminal support highlighting?                   no
  Is foreground color supported?                       no
  Is BMS to generate extended data streams?             no
  Program Symbols                                       no
  Does terminal support field outlining?                no
  Does terminal support SOSI                            no
  Display error messages on the last line?              yes
  Should error messages be intensified?                 yes
  Color for error messages                              no
  Extended highlighting for error messages              no
  Device Type for model entry                          []
  Protection level                                      none
[BOTTOM]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 92. Defining a Remote Printer in AIX



```
 DEF TRAN(REMP) G(R4AD5110)
 OVERTYPE TO MODIFY                                        CICS RELEASE = 0330
  CEDA  ALter
   TRansaction    : REMP
   Group          : R4AD5110
   DEscription  ==>
   PROGram      ==>
   TWasize      ==> 00000              0-32767
   PROFile      ==> DFHCICST
   PArtitionset ==>
   STatus       ==> Enabled            Enabled | Disabled
   PRIMedsize     : 00000              0-65520
   TASKDATALoc  ==> Below              Below | Any
   TASKDATAKey  ==> User               User | Cics
  REMOTE ATTRIBUTES
   DYnamic      ==> No                 No | Yes
   REMOTESystem ==>  R6K2
   REMOTEName   ==>  REMP
   TRProf       ==> DFHCICSS
   Localq       ==> No                 No | Yes
  SCHEDULING
   PRIOrity     ==> 001                0-255
   TClass       ==> No                 No | 1-10
  ALIASES
   Alias        ==>
   TASKReq      ==>
   XTRanid      ==>
   TPName       ==>
                ==>
   XTPname      ==>
                ==>
                ==>
  RECOVERY
   DTimout      ==> No                 No | 1-6800
   Indoubt      ==> Backout            Backout | Commit | Wait
   RESTart      ==> No                 No | Yes
   SPurge       ==> No                 No | Yes
   TPUrge       ==> No                 No | Yes
   DUmp         ==> Yes                Yes | No
   TRACe        ==> Yes                Yes | No
  SECURITY
   RESSec       ==> No                 No | Yes
   Cmdsec       ==> No                 No | Yes
   Extsec         : No                 No | Yes
   TRANsec        : 01                 1-64
   RSl            : 00                 0-24 | Public


                                                              APPLID=SCMCICSA

PF 1 HELP 2 COM 3 END             6 CRSR 7 SBH 8 SFH 9 MSG 10 B 11 SF 12 CNCL
```

Figure 93. Defining a Remote Transaction in MVS



```
                              Add Transaction

    Type or select values in entry fields.
    Press Enter AFTER making all desired changes.

                                                      [Entry Fields]
    * Transaction Identifier                        [REMP]
    * Model Transaction Identifier                   ""
    * Region name                                   [SJA2109I]             +
      Add to database only OR Add and Install        Add AND Install       +
      Group to which resource belongs               []
      Activate the resource at cold start?           yes                   +
      Resource description                          [Transaction Definition]
    * Number of updates                              0
      Protect resource from modification?            no                    +
      Transaction enable status                      enabled               +
      Remote System Identifier                      []
      Remote Transaction Identifier                 []
      Resource Level Security Key                   [public]
      Transaction Level Security Key                [1]
      Type of RSL Checks                             none                  +
      Type of TSL Checks                             internal              +
      Should transaction be dumped on an abend?      no                    +
      First program name                            [REMO]
      SNA TPN profile for APPC listener program     [CICSTPN]
      Is a user conversion template defined?         no                    +
      Is back end of a DTP transaction?              yes                   +
      Enabled at shutdown?                           no                    +
      Transaction Work Area Size                    [0]                    #
      Transaction Priority                          [0]                    #
      Transaction purgeability                       purgeable             +
      Transaction deadlock timeout value (secs)     [0]                    #
      Effect of FORCEPURGE for InDoubt transactions  wait_backout          +
      Should data be converted to uppercase?         no                    +
      SNA modename for this transaction             []
      Locally Queue ATI requests for this transaction? no                  +
      Conversational timeout value (mins)           [0]                    #
      Contexts in which transaction can START       [no_facility|output_ter>

    F1=Help             F2=Refresh          F3=Cancel           F4=List
    F5=Reset            F6=Command          F7=Edit             F8=Image
    F9=Shell            F10=Exit            Enter=Do
```

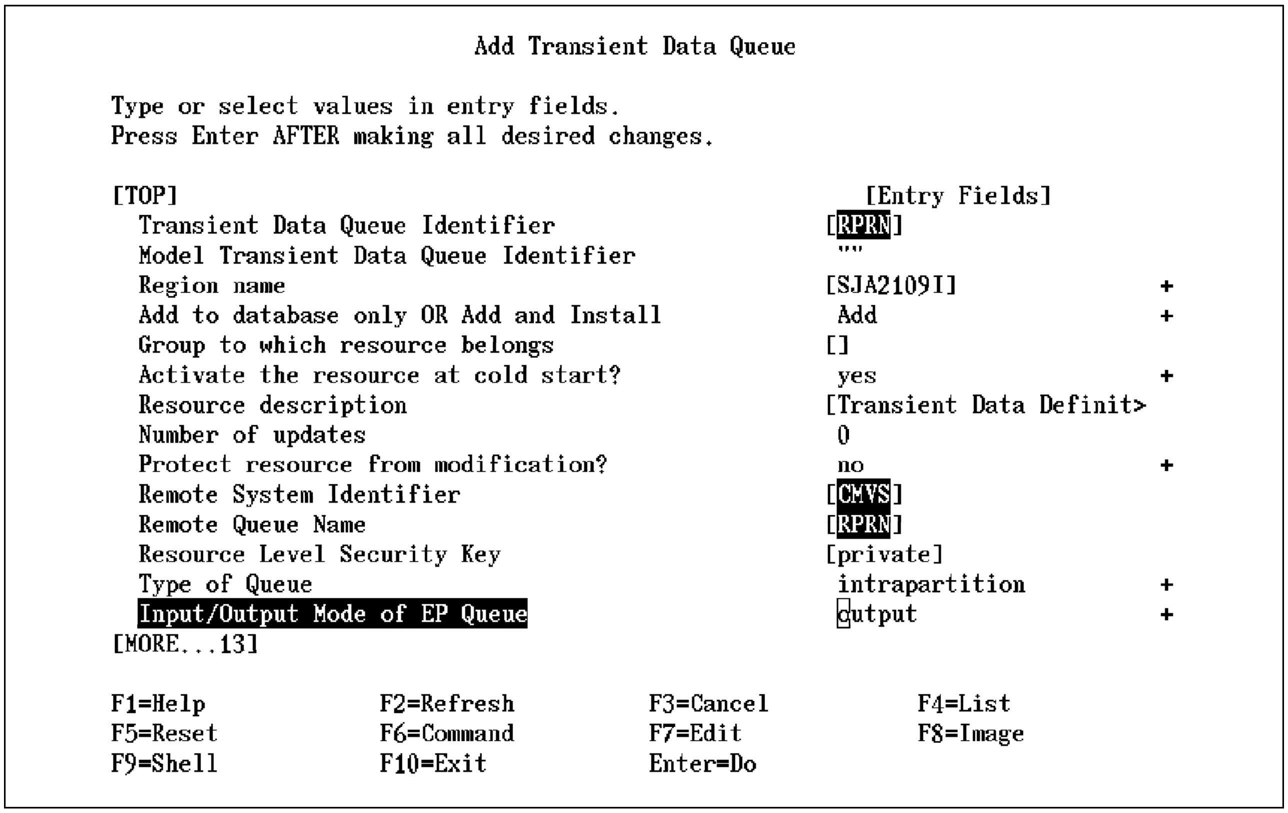
Figure 94. Setting the SNA TPN Profile

You can find out how to define the SNA TPN profile, CICSTPN, in Appendix D, “AIX Definitions” on page 111.

13.1.4 Transient Data Queue with a Trigger Level

You can send symbolic map data structures to a transient data (TD) queue using the WRITEQ TD command. CICS can be made to initiate a print transaction when a specific number of records have been written to the TD queue. To avoid the problem of interleaving the output of several instances of the print transaction on the TD queue, you will have to store all data to be printed by an instance in a single TD queue item. You may also use the ENQ and DEQ commands to get exclusive control of the TD queue, but this method may lead to delays due to enqueueing on the TD resource.

To use this print option in scenario 1, you have to define a destination control table (DCT) entry in CICS/ESA, where you set the name of the transaction to be initiated, the identifier of the printer that is to be its principal facility, and the trigger level at which it is started. You also have to define a remote TD queue on the AIX side, referring to the queue on the host side, as shown in Figure 95. The transaction triggered must be in the same CICS system as the TD queue and the printer terminal. If not, the transaction fails, and you will receive a warning message on the console. You do not have to modify the application to use this option in scenario 1.



```
                          Add Transient Data Queue

    Type or select values in entry fields.
    Press Enter AFTER making all desired changes.

    [TOP]                                                [Entry Fields]
      Transient Data Queue Identifier                  [RPRN]
      Model Transient Data Queue Identifier             ""
      Region name                                      [SJA2109I]            +
      Add to database only OR Add and Install           Add                  +
      Group to which resource belongs                  []
      Activate the resource at cold start?              yes                  +
      Resource description                             [Transient Data Definit>
      Number of updates                                 0
      Protect resource from modification?               no                   +
      Remote System Identifier                         [CMVS]
      Remote Queue Name                                [RPRN]
      Resource Level Security Key                      [private]
      Type of Queue                                     intrapartition       +
      Input/Output Mode of EP Queue                     output               +
    [MORE...13]

    F1=Help           F2=Refresh        F3=Cancel         F4=List
    F5=Reset          F6=Command        F7=Edit           F8=Image
    F9=Shell          F10=Exit          Enter=Do
```

Figure 95. Defining a Remote TD Queue in CICS/6000

13.1.5 Writing to the JES Spool

The CICS spooler interface to the job entry subsystem (JES) component of MVS allows you to access the system spool data sets maintained by JES and exchange data sets with other systems that are connected through a JES remote spooling communication subsystem (RSCS) network. With this facility, you can:

- Retrieve data for a specific MVS external writer name from the local JES spool data set
- Create a data set and write records directly to an output queue of the local JES spool data set, to be printed in the system printer
- Create a data set and write a job stream directly to an input queue of the spool data set, to be submitted for execution
- Send a JES spool data set to a specific remote destination.

As there is no equivalent of the JES spool in the AIX system, you have to modify your application to deal with this limitation. In scenario 1, you have the option of splitting your application, leaving the modules that deal with the JES spool on the host side, while invoking them through distributed program link (DPL) calls.

To submit a job stream for execution, you can convert your job stream to script sentences and write them to the BATCH queue of AIX, as explained in Chapter 11, “Batch Jobs” on page 69.

13.2 Scenario 2

CICS/6000 offers no direct printer support as is offered in the mainframe CICS environment. CICS/6000 also does not support the CICS local copy key. Printers are supported using the cicstermp component. The cicstermp component is similar to the cicsterm 3270 terminal emulator but is not connected to an interactive device, so keyboard input cannot be received.

The cicstermp process is associated with a logical CICS/6000 terminal. CICS application programs send their print data to that terminal, where cicstermp routes printer data streams to a temporary file, which in turn is passed on to an AIX command you specify when you invoke cicstermp.

13.2.1 Defining the CICS Printer Resource

The CICS 3270 Terminal Emulator printer is not autoinstalled; you must define an output-only terminal resource to CICS/6000 before a printer can be attached to your CICS/6000 region.

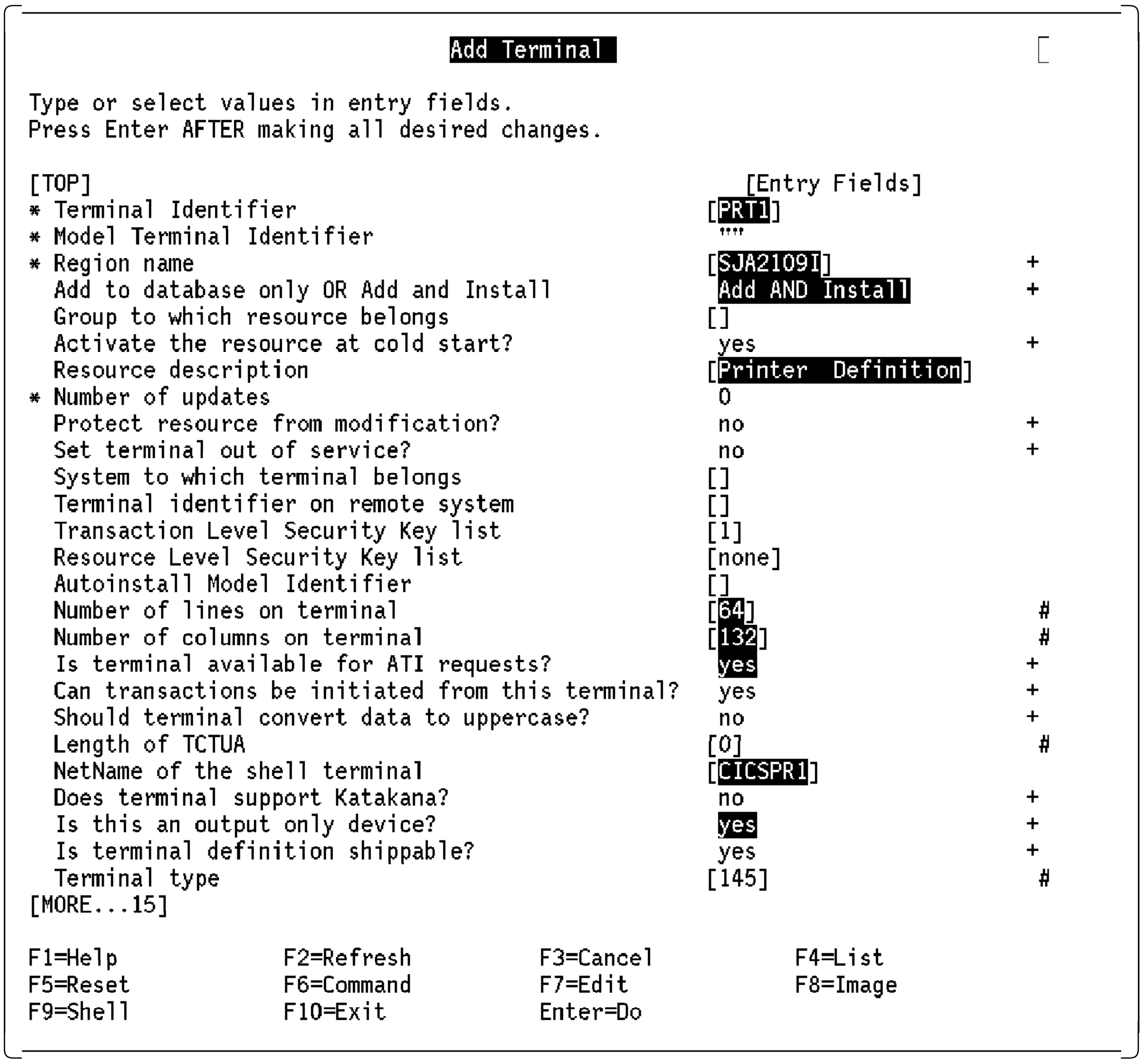
The cicstermp process emulates a fixed-sized printer of 132 columns by 64 lines. You must use these values when defining the resource to CICS/6000. Any other values will result in failure of the installation of the terminal into the run-time system. If your physical printer has different characteristics (for example, it is 80 columns wide), you have to specify such characteristics as a parameter of the `qprt` command.

The terminal must be available for ATI requests, as the printing transaction is asynchronously started by the EXEC CICS START interface.

Figure 96 on page 95 shows the definition of a terminal with a netname of CICSPR1. The netname is used when invoking the cicstermp command.

To create a terminal resource named CICSPR1 through SMIT, enter the fastpath command: `smitty cicsaddwd`.

You should select ▌""▐ when prompted for a model terminal identifier.



```
                              Add Terminal

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

[TOP]                                                  [Entry Fields]
* Terminal Identifier                               [PRT1]
* Model Terminal Identifier                          ""
* Region name                                       [SJA2109I]             +
  Add to database only OR Add and Install            Add AND Install       +
  Group to which resource belongs                   []
  Activate the resource at cold start?               yes                   +
  Resource description                              [Printer  Definition]
* Number of updates                                  0
  Protect resource from modification?                no                    +
  Set terminal out of service?                       no                    +
  System to which terminal belongs                  []
  Terminal identifier on remote system              []
  Transaction Level Security Key list               [1]
  Resource Level Security Key list                  [none]
  Autoinstall Model Identifier                      []
  Number of lines on terminal                       [64]                    #
  Number of columns on terminal                     [132]                   #
  Is terminal available for ATI requests?            yes                   +
  Can transactions be initiated from this terminal?  yes                   +
  Should terminal convert data to uppercase?         no                    +
  Length of TCTUA                                   [0]                     #
  NetName of the shell terminal                     [CICSPR1]
  Does terminal support Katakana?                    no                    +
  Is this an output only device?                     yes                   +
  Is terminal definition shippable?                  yes                   +
  Terminal type                                     [145]                   #
[MORE...15]

F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 96. Adding a Printer Terminal

13.2.2 Starting the cicstermp Process

A cicstermp process is attached to a particular CICS/6000 region. It is invoked manually as shown in Figure 97,

```
cicstermp -n CICSPR1 -P′qprt -c -d lp0′ -r $CICSREGION
```

Figure 97. Invoking cicstermp

where:

- CICSPR1 is our netname for the printer
- `-P′qprt -c -d lp0′` is the AIX command to which cicstermp will pass the temporary file it has created from the printer data stream
- $CICSREGION is the environment variable containing the name of your CICS/6000 region.

13.2.3 BMS Printed Output

Very often you will want printed output (hard copy) as well as, or instead of, the screen images produced by a transaction. You have a choice of methods of producing such output. The method you choose depends on your requirements. This section describes one method, the asynchronous page build transaction.

A 3270 printer contains a page buffer. BMS moves data into this page buffer when instructed to do so by SEND MAP or SEND CONTROL commands. The page buffer is printed only when BMS receives a SEND MAP or SEND CONTROL command containing the PRINT option. Likewise, it is erased only if BMS receives a SEND MAP or SEND CONTROL command that specifies the ERASE option. These properties of the printer make it possible for a program to build a single page of printed output from a series of maps.

Two ways of printing a page built from multiple maps are:

- Using the interval control START command

  You use the START command to initiate a secondary CICS task. This is a print task if the TERMID option of the command names a printer as its principal facility. Your initial transaction can pass data to the print task by specifying the FROM and LENGTH options of the START command. If the primary transaction has already created a series of output data structures in the FROM area, the secondary transaction can map the data to the printer buffer, then initiate printing using a BMS SEND with the PRINT option.

- Using a transient data queue with a trigger level

  You can send symbolic map data structures to a transient data queue using the WRITEQ command. CICS can be made to initiate a print transaction when a specific number of records have been written to the queue. The name of the transaction to be initiated, the identifier of the printer that is to be its principal facility, and the trigger level at which it is started are defined in the transient data definitions (TDD). Note, however, that output from several instances of your transaction may be interleaved on the transient data queue. This can be avoided if all data to be printed by an instance of your transaction is stored in a single transient data queue item. Alternatively, each instance of your transaction can get exclusive control of the transient data queue by ENQ and DEQ commands.

Appendix A. Migration Tidbits

In this appendix we present some valuable bits of information that cannot be found in any other part of the book. In particular we mention some tools and software products that we did not need or use in our project, but that may be useful to you in your project.

A.1 Programming Languages

CICS/6000 supports the following programming languages:

- COBOL
- C.

DB2/6000 supports the programming languages:

- COBOL
- C
- FORTRAN.

There is no assembler language support and currently no PLI support for CICS/6000 nor DB2/6000. Programs written in these languages will have to be rewritten. Micro-Processor Services, Inc., has a set of offerings for language translation. These include PLI to C, COBOL to C, ASM370 to C, and ASM370 to COBOL. You can contact them at:

```
Micro-Processor Services, Inc.
92 Stone Hurst Lane
Dix Hills, New York  11746 USA
+1-516-499-4461 (phone)
```

A.2 Databases

In general you will not encounter major problems migrating from any relational database to DB2/6000. It is far more problematic to migrate from a hierarchical or network database system to DB2/6000.

SWS Software GmbH offers a tool called *HIREL***. This tool will help you convert a complete application from Information Management System/Database (IMS/ESA DB) and Data Language I (DLI) to DB2 and SQL. SWS also offers tools for converting VSAM files and programs to DB2 and SQL. You can reach SWS at:

```
SWS Software GmbH
Karlsruher Strasse 38
D-75179 Pforzheim
Germany

+49-7231-37-88-0 (phone)
+49-7231-37-88-88 (fax)
```

You may find that you have to re-create certain objects of your DB2/6000 database. DB2/6000 currently does not provide any tools for generating the DDL create statements from existing objects in the catalog. An *IBM Internal Use Only* tool provides this function and a lot more. It is called *DB2TOOLS* and can be requested from the AIXTOOLS disk. This package was created by Kirk Beaty and contains the following tools:

- DBopen - Reduce Application Database Connection Overhead
- genDDL - Generate Database Table DDL
- genDCL - Generate C Language DCL Include Files
- listcols - Lists Column Definitions for Database Tables/Views
- Prune - Prune Rows from a Table based on Size and Age Limits
- reorgscan - Automated REORGCHK Report Processing.

A.3 CICS API

There are slight differences between the CICS API offered in the different CICS systems. *Orbit* has developed a tool to examine programs to find out how much work will be involved to migrate from one CICS platform to another. *IBS* has developed a similar tool. You can reach these companies at:

```
Orbit
Alexandra Buildings
28 Queen St,
Lincoin Square,
Manchester
M2 5LF
UK
+44-61-832-9506 (phone)

IBS
2 Bath St
London
EC1V 9LB
UK
+44-71-454-9822 (phone)
+44-71-454-9821 (fax)
```

The Orbit tool also deals with DB2 and COBOL issues.

CICS/6000 does not support the macro-level interface. The SupportPac tool CA20, CICS/VSE Application Migration Aid (AMA), can assist you in the conversion from macro-level to command-level CICS calls.

A.4 Information Highways

If you have Internet access you can retrieve information on the CICS/6000 system from the World Wide Web (WWW) server, *http://cics.aix.dfw.ibm.com*. You will find questions, answers, and other information at this location. You can access the WWW server using the mosaic application. Mosaic WWW Browser is available from the anonymous ftp server, ftp.ncsa.uiuc.edu, in both source and AIX binary forms.

The IBM forums are also invaluable sources of information. They provide an unofficial means of posting questions and having them answered with minimum delay directly by IBM specialists. Customers can access the following forums:

- DB26000 CFORUM

- CICS6000 CFORUM
- VTAM CFORUM
- TCPIP CFORUM
- TCPIPMVS CFORUM.

Contact your local IBM technical professional to learn how to access the IBM customer forums.

Please remember that to obtain official answers concerning any IBM program products you must use the official support channels.

You can also find a lot of useful information on the Internet. The following news groups discuss the products you can use when downsizing to AIX:

- comp.protocols.tcp-ip
- DB2-L@AUVM.AMERICAN.EDU
- CICS-L@UGA.BITNET
- INFO-ENCINA@TRANSARC.COM
- sig-dce@osf.org.

You can also find frequently asked question (FAQ) files, which may be of help to you in the initial phase of your downsizing project, for example, on:

- comp.unix.aix
- comp.protocols.tcp-ip.

You will need access to the Internet to participate in the news groups or to request the FAQ files.

A.5 Hursley SupportPacs

There is a wide range of professional services which IBM is able to offer customers to support the CICS and MQSeries* range of products. A series of tools and other materials assist the IBM professional to deliver services which are effective, efficient, consistent, and replicable, as well as being tailored to customers′ individual needs. These tools and materials are called SupportPacs.

A list of SupportPac titles is shown below. These titles will give an indication of the type of service which customers can request from their IBM Representative. SupportPacs are not, in themselves, orderable items from IBM. Materials, methods, and tools contained in the SupportPacs are to aid the delivery of services by IBM for customers′ benefit. Such services are arranged through customers′ account representatives, who may contact IBM Services Segment, based at IBM UK Laboratories Ltd, Hursley Park, England, for help in delivery if required. (EUROSERV at WINVMD, UK Telephone (0)962-816211).

The following list shows you the available SupportPacs.

- CA10: CICS Application Development Support
- CA17: Capacity Planning for CICS Cross Platform Migrations
- CA18: COBOL II Application Programming in a CICS Environment
- CA19: Application Design for CICS/DB2

- CA20: CICS/VSE Application Migration Aid
- CA31: CICS OS/2 - A REXX to CICS OS/2 Interface (RXCICS)
- CA32: CICS OS/2 - Purge Task Transaction
- CA60: CICS/6000 Application Porting Pac
- CA64: CICS/6000 External Presentation Interface Examples
- CD12: CICS/ESA Front End Programming Interface (FEPI) Appreciation Pac
- CD61: CICS/6000 - DB2/6000 Exploitation
- CD62: CICS/6000 - ECI Prototype
- CE00: CICS Cross Platform Intersystem Communications (ISC) Examples
- CE10: CICS/ESA Migration Feasibility Study
- CE11: CICS/ESA Migration Workbook
- CE12: CICS/ESA DBCTL Implementation
- CE14: CICS/MVS External Security
- CE17: CICS/ESA Shutdown Assist Program
- CE31: CICS TCP/IP - CICS OS/2 - CICS Gateway
- CE34: CICS/MVS File Transfer Code
- CE42: CICS/400 Migration Feasibility
- CE61: CICS/6000 Migration Feasibility
- CE62: CICS/6000 System Distribution Guide
- CE63: CICS/6000 On-Line Transaction Processing with AIX
- CE64: CICS/6000 Configuration Tools
- CE68: CICS/600 System Configuration Aid (SCA) Tool
- CH10: CICS Health Check
- CH13: CICS/MVS File Transfer Program Fix
- CP00: CICS/MVS/ESA Performance Management
- CP11: Tuning the CICS/DB2 Interface
- CP31: CICS OS/2 Performance Analyser/2
- CP62: High Performance Read Only Access to Key Sequenced Data in a CICS/6000 Environment
- CR60: CICS/6000 Backup and Recovery
- CS11: CICS AO/MVS Implementation Service
- CS17: CICS/MVS - CICS Interdependencies Utility
- CS19: CICSPlexSM - A Real Time Analysis Cookbook
- CS40: CICS/400 Control Region Creation Aid for Inter System Communication
- CS60: Changing Hostname or TCP/IP Address in a DCE Cell with CICS/6000
- CS61: CICS/6000 Change Password (CPWD)
- CS62: CICS/6000 Terminal Usage Monitor (CWHO)
- CS63: Starting CICS/6000 Printers
- MA10: MQSeries MQM MVS/ESA ISPF Utilities

- MD10: MQSeries in the MVS/ESA Batch Environment
- MD11: MQSeries in the MVS/ESA Batch Environment Performance Guide
- MS10: MQSeries/ESA Exerciser

This list is dated June 1994. New SupportPacs are being added frequently. SupportPacs are held on the TXPPACS disk managed by TOOLS at WINVMB. IBM readers can obtain an up to date list and further information, or download individual Pacs by using the following tools commands on IBM internal systems. Figure 98 shows the command to request a list and a brief description of available SupportPacs.

```
TOOLS SENDTO WINVMB TOOLS TXPPACS GET TXPSUM TEXT
```

Figure 98. Request a List of Available SupportPacs

Figure 99 shows you how to request a SupportPac, where nnnn is the name of the SupportPac, for example, CA10.

```
TOOLS SENDTO WINVMB TOOLS TXPPACS GET nnnn PACKAGE
```

Figure 99. Request a Specific SupportPac

Appendix B. Integrating CICS/6000 and DB2/6000

In this appendix we briefly describe the steps needed to integrate CICS/6000 with DB2/6000 using the XA interface. We assume you have successfully installed and configured both products on your system. We divide the steps into two parts:

- DB2/6000 XA configuration
- CICS/6000 XA configuration.

B.1 DB2/6000 XA Configuration

The items you need to consider in the implementation of the XA interface are:

1. Shared object
2. XA open string
3. Resource manager switch
4. Database privileges.

B.1.1 Shared Object

You need to create a DB2/6000 shared object code. This is shared code loaded into memory once in the shared library segment and shared by all processes that reference it. The CICS/6000 COBOL run time, CICS/6000 C transactions, and the Switch Load File need to reference the DB2/6000 shared object at run time.

The steps involved in creating the shared object are:

1. AIX login as root
2. cd /usr/lpp/db2_01_01_0000/lib
3. ar -xv libdb2.a
4. mv shr.o db2.o

Symbolic Links

If, after installing DB2/6000, you used the db2ln script to create symbolic links from /usr to the DB2/6000 library and include files, you need to define another link as follows:

```
ln -s /usr/lpp/db2_01_01_0000/lib/db2.o /usr/lib/db2.o
```

B.1.2 XA Open String

The XA open string identifies each database to CICS/6000 and has the syntax shown in Figure 100:

```
database_alias<username,password>
```

Figure 100. Database Identification

The field values are:

database_alias is the database name unless you explicitly cataloged an alias name after creating the database.

username is a valid AIX userid and is optional. It is used to provide authentication information to the database if the database is set up with authentication=SERVER.

password This is the password for the username.

Note

Because DB2/6000 uses a database name in the XA Open String you will need to:

- Create the database before region startup.
- Define an XA Open String for each database that you use in a CICS/6000 application.

B.1.3 Resource Manager Switch

The C source file for the the Switch Load File for DB2/6000 can be found in the */usr/lpp/cics/v1.1/src/examples/xa* directory. It is called db2xa.c.

Build the Switch Load File object by issuing the command shown in Figure 101.

```
make -f db2xa.mk
```

Figure 101. Make Command for Switch Load File

The makefile can also be found in the */usr/lpp/cics/v1.1/src/exa mples/xa* directory. The make command creates a loadable file called db2xa. You must copy this file to your CICS/6000 region as shown in Figure 102.

```
cp db2xa /var/cics_regions/<cicsregion>/bin/db2xa
```

Figure 102. Copying db2xa to the Region

where <cicsregion> is the name of your region.

B.1.4 Database Privileges

At region startup the CICS application server makes an initial connection to the resource manager, in this case DB2/6000, using the XA Open String. The AIX userid used is **cics**. Therefore the **cics** userid should have the privileges granted to access the database by the commands shown in Figure 103, where *<database name>* is the name of your database.

```
db2 connect to <database name>
db2 grant connect on database to cics
```

Figure 103. Granting Database Privileges

B.2 CICS/6000 XA Configuration

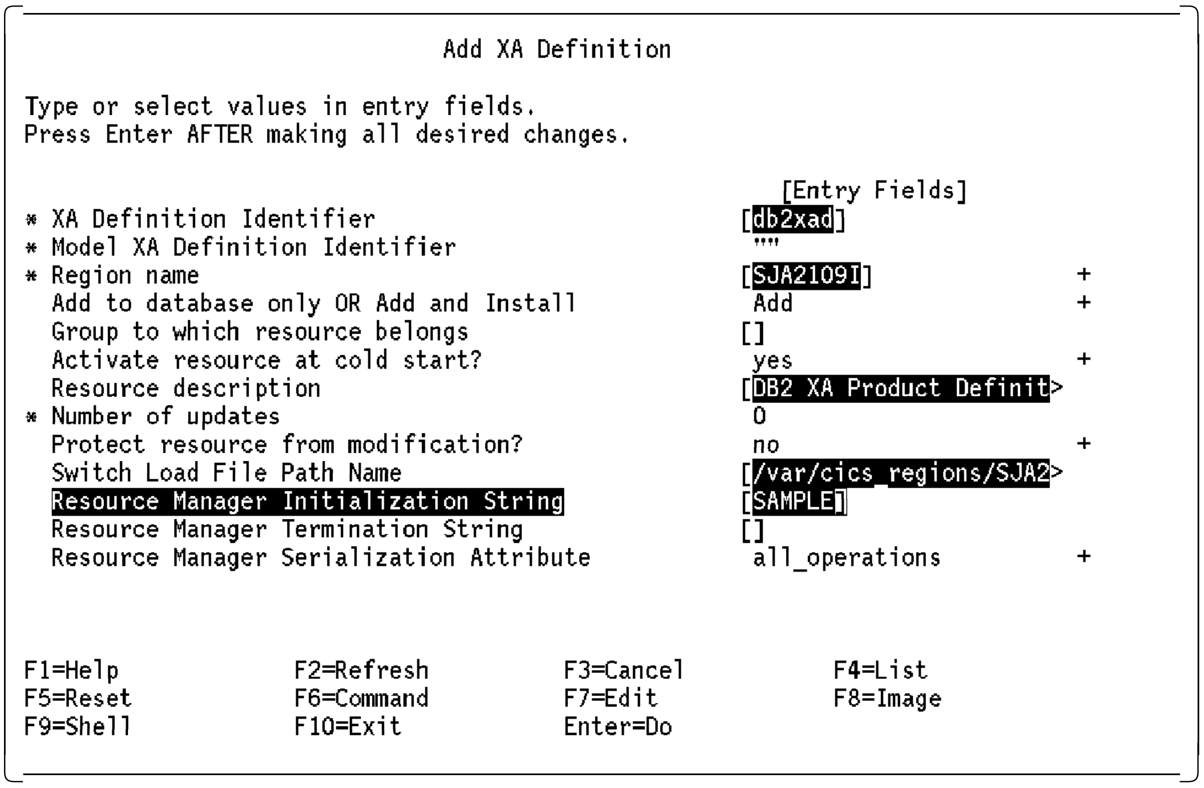
The steps required for the CICS/6000 setup are as follows:

- XA resource definition
- CICS/6000 region environment setup
- CICS/6000 region startup

B.2.1 XA Resource Definition

You need to define DB2/6000 as a resource manager to CICS/6000. Your AIX userid must be a member of the **cics** group in order to access and modify CICS/6000 resources.

Enter the fastpath command: `smitty cicsaddxad` to get a panel similar to that in Figure 104.



```
                              Add XA Definition

Type or select values in entry fields.
Press Enter AFTER making all desired changes.

                                                        [Entry Fields]
* XA Definition Identifier                            [db2xad]
* Model XA Definition Identifier                       ""
* Region name                                         [SJA2109I]             +
  Add to database only OR Add and Install              Add                   +
  Group to which resource belongs                     []
  Activate resource at cold start?                     yes                   +
  Resource description                                [DB2 XA Product Definit>
* Number of updates                                    0
  Protect resource from modification?                  no                    +
  Switch Load File Path Name                          [/var/cics_regions/SJA2>
  Resource Manager Initialization String              [SAMPLE]
  Resource Manager Termination String                 []
  Resource Manager Serialization Attribute             all_operations        +



F1=Help             F2=Refresh          F3=Cancel           F4=List
F5=Reset            F6=Command          F7=Edit             F8=Image
F9=Shell            F10=Exit            Enter=Do
```

Figure 104. DB2/6000 XA Definition for CICS/6000

The highlighted values are as follows:

- The XA Definition Identifier is unique for every XA Definition.
- The Resource Description is a comment line.
- The Switch Load File Path Name is `/var/cics_regions/<cicsregion>/bin/db2xa`, where <cicsregion> is the name of your region. This Switch Load File is the one you created in B.1.3, "Resource Manager Switch" on page 104.
- SAMPLE is the XA open string. No username or password has been provided. This implies that the database was created with **authentication=client**.

B.2.2 CICS/6000 Region Environment Setup

Every CICS/6000 region has an environment file called */var/cics_regions/<cicsregion>/environment*. This file contains the environment variables that need to be set up at region startup. The environment variable for DB2/6000 is :

DB2INSTANCE Defines the instance for CICS/6000 to refer to.

You need to insert this variable and all others related to DB2/6000 into the environment file. The syntax for an entry in the file is:

DB2INSTANCE=db2

where **db2** is the instance name.

B.2.3 CICS/6000 Region Startup

Now that you have configured all of the required components, it is time to bring up the CICS/6000 region. Perform a cold start (smitty cicscoldstart).

Check the **console.msg** file in the */var/cics_regions/< cicsregion>* directory for the successfull startup of the region. You should see something similar to that shown in Figure 105.

```
ERZ8006I/0806 06/09/94 09:32:24 SJA2109I     : XA_OPEN succeeded:
Application Server  6 connected to ′DB2/6000′ using XA_OPEN string
 ′SAMPLE′
ERZ1020I/0068 06/09/94 09:32:25 SJA2109I     : *** CICS/6000 startup
is complete ***
```

Figure 105. Sample console.msg

Appendix C. MVS Definitions

You need the following definitions in the MVS/ESA environment to define the link between CICS/ESA and CICS/6000; that is, an ISC connection through VTAM, implementing the LU 6.2 protocol:

- RDO connection
- RDO sessions
- SIT parameters
- VTAM APPL and CDRSC
- VTAM PU-LU
- VTAM modetab entry.

C.1 RDO Connection

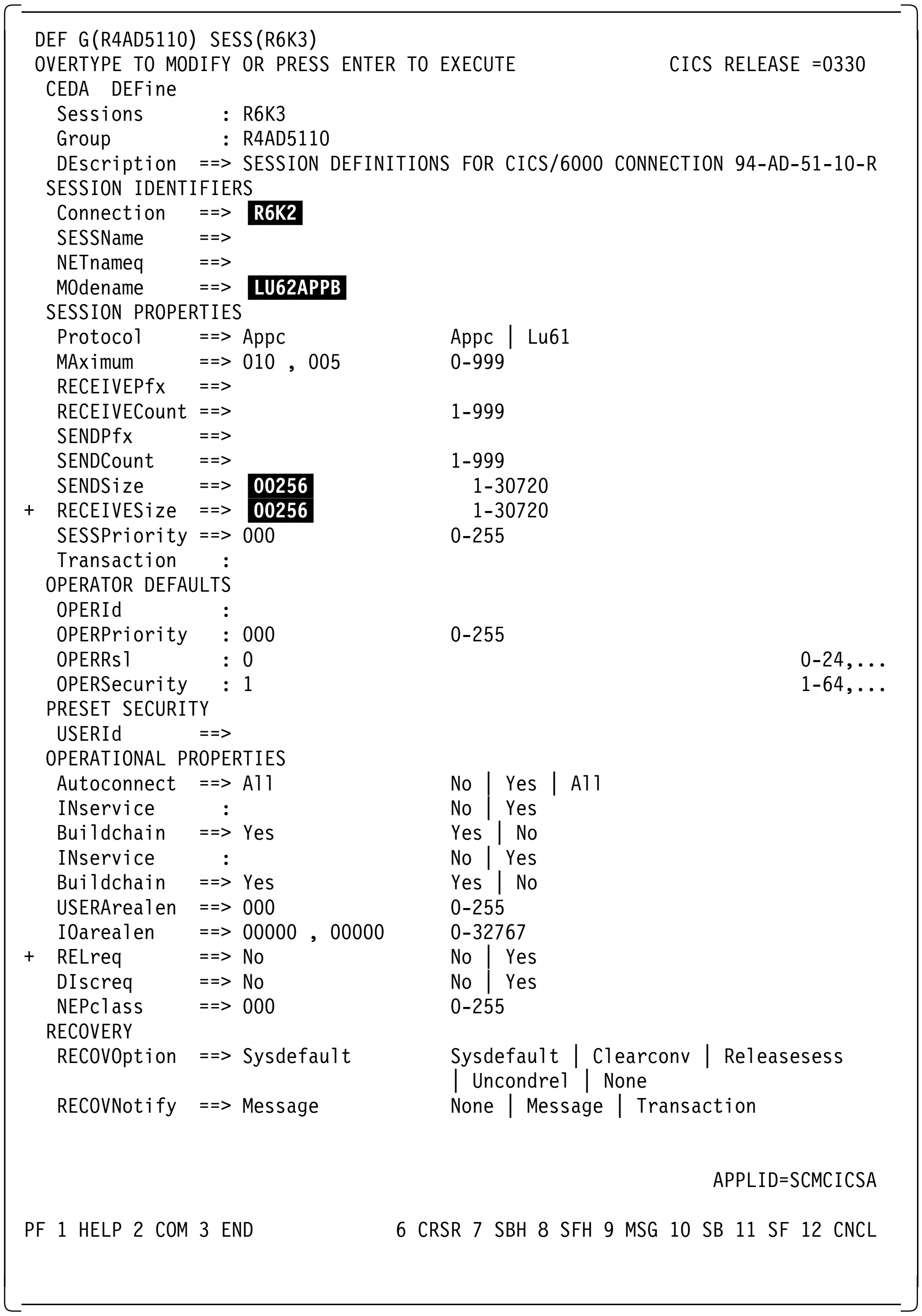
Figure 106 shows the RDO connection definition used in this project.

```
 DEF G(R4AD5110) CONN(R6K2)
 OVERTYPE TO MODIFY OR PRESS ENTER TO EXECUTE              CICS RELEASE =0330
  CEDA  DEFine
   Connection     :  R6K2
   Group          : R4AD5110
   DEscription  ==> DEFINITION OF CICS/6000 CONNECTION FOR 94-AD-51-10-R
  CONNECTION IDENTIFIERS
   Netname      ==>  SJA2109I
   INDsys       ==>
  REMOTE ATTRIBUTES
   REMOTESystem ==>
   REMOTEName   ==>
  CONNECTION PROPERTIES
   ACcessmethod ==> Vtam             Vtam | IRc | INdirect | Xm
   Protocol     ==> Appc             Appc | Lu61
   SInglesess   ==> No               No | Yes
   DAtastream   ==> User             User | 3270 | SCs | STrfield | Lms
   RECordformat ==> U                U | Vb
  OPERATIONAL PROPERTIES
+  AUtoconnect  ==> All              No | Yes | All
   INService    ==> Yes              Yes | No
  SECURITY
   SEcurityname ==>
   ATtachsec    ==> Local            Local | Identify | Verify | Persistent
                                     | Mixidpe
   BINDPassword ==>                  PASSWORD NOT SPECIFIED
   BINDSecurity ==> No               No | Yes

                                                               APPLID=SCMCICSA

PF 1 HELP 2 COM 3 END            6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 106. RDO Connection Definition (CICS/ESA)

C.2 RDO Sessions

Figure 107 shows the RDO sessions definition used in this project.



```
 DEF G(R4AD5110) SESS(R6K3)
 OVERTYPE TO MODIFY OR PRESS ENTER TO EXECUTE              CICS RELEASE =0330
  CEDA  DEFine
   Sessions       : R6K3
   Group          : R4AD5110
   DEscription  ==> SESSION DEFINITIONS FOR CICS/6000 CONNECTION 94-AD-51-10-R
  SESSION IDENTIFIERS
   Connection   ==>  R6K2
   SESSName     ==>
   NETnameq     ==>
   MOdename     ==>  LU62APPB
  SESSION PROPERTIES
   Protocol     ==> Appc               Appc | Lu61
   MAximum      ==> 010 , 005          0-999
   RECEIVEPfx   ==>
   RECEIVECount ==>                    1-999
   SENDPfx      ==>
   SENDCount    ==>                    1-999
   SENDSize     ==>  00256               1-30720
+  RECEIVESize  ==>  00256               1-30720
   SESSPriority ==> 000                0-255
   Transaction    :
  OPERATOR DEFAULTS
   OPERId         :
   OPERPriority   : 000                0-255
   OPERRsl        : 0                                                 0-24,...
   OPERSecurity   : 1                                                 1-64,...
  PRESET SECURITY
   USERId       ==>
  OPERATIONAL PROPERTIES
   Autoconnect  ==> All                No | Yes | All
   INservice      :                    No | Yes
   Buildchain   ==> Yes                Yes | No
   INservice      :                    No | Yes
   Buildchain   ==> Yes                Yes | No
   USERArealen  ==> 000                0-255
   IOarealen    ==> 00000 , 00000      0-32767
+  RELreq       ==> No                 No | Yes
   DIscreq      ==> No                 No | Yes
   NEPclass     ==> 000                0-255
  RECOVERY
   RECOVOption  ==> Sysdefault         Sysdefault | Clearconv | Releasesess
                                       | Uncondrel | None
   RECOVNotify  ==> Message            None | Message | Transaction


                                                                APPLID=SCMCICSA

PF 1 HELP 2 COM 3 END           6 CRSR 7 SBH 8 SFH 9 MSG 10 SB 11 SF 12 CNCL
```

Figure 107. RDO Sessions Definition (CICS/ESA)

C.3 SIT Parameters

Figure 108 shows the SIT parameters required to enable the ISC communication.

```
      ...
    ISC=YES,                  * INTERSYSTEM COMMUNICATION X
    SYSIDNT=CICA,             * SYSTEM IDENTIFICATION     X
    APPLID= SCMCICSA ,        * INTERSYSTEM COMMUNICATION X
      ...
```

Figure 108. SIT Parameters (CICS/ESA)

C.4 VTAM APPL and CDRSC

Figure 109 shows the VTAM APPL and CDRSC definitions we used to establish the connection between MVS and AIX, using the VM system, SCG20, as a gateway for cross-domain resources.

```
  ...
SJA2109A CDRSC CDRM=SCG20,ISTATUS=ACTIVE
SJA2109B CDRSC CDRM=SCG20,ISTATUS=ACTIVE
 SJA2109I  CDRSC CDRM= SCG20 ,ISTATUS=ACTIVE
SJA2109J CDRSC CDRM=SCG20,ISTATUS=ACTIVE
  ...

SCMCICSA  APPL   AUTH=(ACQ),EAS=1200,ACBNAME= SCMCICSA ,PARSESS=YES,  X
               MODETAB=SCMODIMS

  ...
```

Figure 109. VTAM APPL and CRDSC Definitions (MVS/ESA)

PARSESS must be set to YES to allow LU6.2 parallel sessions.

C.5 VTAM PU-LU

Figure 110 on page 110 shows the VTAM PU and LU definitions of the RISC System/6000 for enabling the SNA communication. These definitions are actually placed in VTAM/VM.

```
       ...
 SJA2109     PU ADDR=01,                                              +
              IDBLK=05D,IDNUM= A2109 ,                                +
              ANS=CONT,DISCNT=NO,                                     +
              IRETRY=NO,ISTATUS=ACTIVE,                               +
              MAXDATA=265,MAXOUT=1,                                   +
              MAXPATH=1,                                              +
              PUTYPE=2,SECNET=NO,                                     +
              MODETAB=POKMODE,DLOGMOD=DYNRMT,                         +
              USSTAB=USSRDYN,LOGAPPL=SCGVAMP,                         +
              PACING=1,VPACING=2
*
SJA2109A    LU LOCADDR=2
 SJA2109I    LU LOCADDR=0,DLOGMOD= LU62APPB
      ...
```

Figure 110. VTAM PU-LU Definitions (VM/ESA)

C.6 VTAM MODETAB Entry

Figure 110 shows the VTAM MODEENT you have to code to enable the SNA link. The send and receive rusizes must be the same. To calculate the rusize, you take the first digit and multiply it by 2 to the power of the second digit.

```
        ...
LU62     MODETAB
LU62APPB MODEENT LOGMODE= LU62APPB ,                                  +
              TYPE=0,                                                 +
              FMPROF=X'13',                                           +
              TSPROF=X'07',                                           +
              PRIPROT=X'B0',                                          +
              SECPROT=X'B0',                                          +
              COMPROT=X'50B5',                                        +
              RUSIZES=X'8585', SEND/RECV       8 * 2· =  256          +
              PSERVIC=X'060200000000000000002F00'
        ...
```

Figure 111. VTAM MODETAB Entry (MVS/ESA)

Appendix D. AIX Definitions

You need the following definitions in the CICS/6000 environment to define the link between CICS/MVS and CICS/6000; that is, an ISC connection through VTAM, implementing the LU 6.2 protocol.

- CICS/6000 connection
- SNA
- Encina PPC Gateway
- DCE.

D.1 CICS/6000 Connection

Two resources have to be defined in the connection definitions (CD) to define the connection between CICS/6000 and the system to which it is to connect. One is for the PPC Gateway and the other is for the MVS connection. Figure 112 and Figure 113 on page 112 show the full listing from the */var/cics_regions/SJA2109I/database/CD/CD.stanza* file.

```
GWYN:
GroupName=""
ActivateOnStartup=yes
ResourceDescription="Communications Definition"
AmendCounter=0
Permanent=no
AllocateTimeout=0
TSLKeyMask=none
RSLKeyMask=none
RemoteSysType=GWY
RemoteLUName="GWYN"
GatewayName=""
RemoteNetworkName=""
SNAConnectName=""
RemoteCodePageTR="IBM-850"
DCECell="/.:/cics/ppc"
GatewayPrincipal=""
RemoteSysSecurity=local
LinkUserId=""
RemoteSysEncrypt=none
OutboundUserIds=not_sent
InService=yes
```

Figure 112. Gateway Definition for CICS/6000 Connection Definition

```
CMVS:
GroupName=””
ActivateOnStartup=yes
ResourceDescription=″Communications Definition″
AmendCounter=0
Permanent=no
AllocateTimeout=0
TSLKeyMask=none
RSLKeyMask=none
RemoteSysType=SNA
RemoteLUName=″SCMCICSA″
GatewayName=″GWYN″
RemoteNetworkName=″USIBMSC″
SNAConnectName=″SJA2109I″
RemoteCodePageTR=″IBM-037″
DCECell=″/.:/″
GatewayPrincipal=””
RemoteSysSecurity=local
LinkUserId=””
RemoteSysEncrypt=none
OutboundUserIds=not_sent
InService=yes
```

Figure 113. Definition for Connection to SCMCICSA

The PPC Gateway definition (GWYN with RemoteSysType=GWY) contains little information other than DCE routing to the server.

The other definition (CMVS with RemoteSysType=SNA) is for the MVS connection. The name, CMVS, is referenced in the CICS/6000 region as the remote SYSID. Note that the SNAConnectName should reflect the profile name for the **LU 6.2 Side Information** profile in the SNA definitions.

D.2 SNA

Figure 114 through Figure 123 on page 116 show the SNA definitions used during the project for communication between CICS/MVS and CICS/6000.

```
sna:
   prof_name                                     = ″sna″
   max_sessions                                  = 200
   max_conversations                             = 200
   restart_action                                = once
   rrm_enabled                                   = no
   dynamic_inbound_partner_lu_definitions_allowed = yes
   standard_output_device                        = ″/dev/console″
   standard_error_device                         = ″/var/sna/sna.stderr″
   nmvt_action_when_no_nmvt_process              = reject
   comments                                      = ””
```

Figure 114. SNA Node Profile

```
control_pt:
   prof_name                                  = "node_cp"
   xid_node_id                                = 0x071a2109
   network_name                               = "USIBMSC"
   control_pt_name_alias                      = "SJA2109"
   control_pt_name                            = "SJA2109"
   control_pt_node_type                       = appn_network_node
   max_cached_trees                           = 500
   max_nodes_in_topology_database             = 500
   route_addition_resistance                  = 128
   comments                                   = ""
```

Figure 115. Control Point Profile

```
local_lu_lu6.2:
   prof_name                                  = "SJA2109I"
   local_lu_name                              = "SJA2109I"
   local_lu_alias                             = "SJA2109I"
   local_lu_dependent                         = no
   local_lu_address                           =
   sscp_id                                    = *
   link_station_prof_name                     = ""
   conversation_security_list_profile_name    = ""
   comments                                   = ""
```

Figure 116. LU 6.2 Local LU Profile

```
partner_lu6.2:
   prof_name                                  = "SCMCICSA"
   fq_partner_lu_name                         = "USIBMSC.SCMCICSA"
   partner_lu_alias                           = "SCMCICSA"
   session_security_supp                      = no
   parallel_session_supp                      = yes
   conversation_security_level                = already_verified
   comments                                   = ""
```

Figure 117. LU 6.2 Partner LU Profile

```
partner_lu6.2_location:
   prof_name                                  = "SCMCICSA"
   fq_partner_lu_name                         = "USIBMSC.SCMCICSA"
   fq_partner_owning_cp_name                  = "USIBMSC.SCG20"
   local_node_is_network_server_for_len_node  = yes
   fq_node_server_name                        = ""
   comments                                   = ""
```

Figure 118. LU 6.2 Partner LU Location Profile

```
side_info:
   prof_name                                   = "SJA2109I"
   local_lu_or_control_pt_alias                = "SJA2109I"
   partner_lu_alias                            = "SCMCICSA"
   fq_partner_lu_name                          = ""
   mode_name                                   = "LU62APPB"
   remote_tp_name_in_hex                       = no
   remote_tp_name                              = ""
   comments                                    = ""
```

Figure 119. LU 6.2 Side Information Profile

```
local_tp:
   prof_name                                   = "CICSTPN"
   tp_name                                     = "DummyTPN"
   tp_name_in_hex                              = no
   pip_data_present                            = no
   pip_data_subfields_number                   = 0
   conversation_type                           = either
   sync_level                                  = all
   resource_security_level                     = none
   resource_access_list_profile_name           = ""
   full_path_tp_exe                            = "/usr/lpp/sna"
   multiple_instances                          = no
   user_id                                     = 100
   server_synonym_name                         = ""
   restart_action                              = once
   communication_type                          = signals
   ipc_queue_key                               = 0
   standard_input_device                       = "/dev/console"
   standard_output_device                      = "/dev/console"
   standard_error_device                       = "/dev/console"
   comments                                    = ""
```

Figure 120. LU 6.2 TPN Profile

```
link_station_token_ring:
   prof_name                                     = "LINKVTAM"
   use_control_pt_xid                            = yes
   xid_node_id                                   = "*"
   sna_dlc_profile_name                          = "LINKVTAM"
   stop_on_inactivity                            = no
   time_out_value                                = 0
   LU_registration_supported                     = no
   LU_registration_profile_name                  = ""
   link_tracing                                  = no
   trace_format                                  = long
   access_routing_type                           = link_address
   remote_link_name                              = ""
   remote_link_address                           = 0x400008210200
   remote_sap                                    = 0x04
   verify_adjacent_node                          = no
   net_id_of_adjacent_node                       = ""
   cp_name_of_adjacent_node                      = ""
   xid_node_id_of_adjacent_node                  = "*"
   node_type_of_adjacent_node                    = learn
   solicit_sscp_sessions                         = yes
   call_out_on_activation                        = yes
   activate_link_during_system_init              = no
   activate_link_on_demand                       = no
   cp_cp_sessions_supported                      = yes
   cp_cp_session_support_required                = no
   adjacent_node_is_preferred_server             = no
   initial_tg_number                             = 0
   restart_on_normal_deactivation                = no
   restart_on_abnormal_deactivation              = no
   restart_on_activation                         = no
   TG_effective_capacity                         = 4300800
   TG_connect_cost_per_time                      = 0
   TG_cost_per_byte                              = 0
   TG_security                                   = nonsecure
   TG_propagation_delay                          = lan
   TG_user_defined_1                             = 128
   TG_user_defined_2                             = 128
   TG_user_defined_3                             = 128
   comments                                      = ""
```

Figure 121. Token-Ring Link Station Profile

```
sna_dlc_token_ring:
   prof_name                                   = "LINKVTAM"
   datalink_device_name                        = "tok0"
   force_timeout                               = 120
   user_defined_max_i_field                    = no
   max_i_field_length                          = 30729
   max_active_link_stations                    = 100
   num_reserved_inbound_activation             = 0
   num_reserved_outbound_activation            = 0
   transmit_window_count                       = 16
   dynamic_window_increment                    = 1
   retransmit_count                            = 8
   receive_window_count                        = 8
   priority                                    = 0
   inact_timeout                               = 48
   response_timeout                            = 4
   acknowledgement_timeout                     = 1
   link_name                                   = ""
   local_sap                                   = 0x04
   retry_interval                              = 60
   retry_limit                                 = 20
   dynamic_link_station_supported              = yes
   trace_base_listen_link_station              = no
   trace_base_listen_link_station_format       = long
   dynamic_lnk_solicit_sscp_sessions           = yes
   dynamic_lnk_cp_cp_sessions_supported        = yes
   dynamic_lnk_cp_cp_session_support_required  = no
   dynamic_lnk_TG_effective_capacity           = 4300800
   dynamic_lnk_TG_connect_cost_per_time        = 0
   dynamic_lnk_TG_cost_per_byte                = 0
   dynamic_lnk_TG_security                     = nonsecure
   dynamic_lnk_TG_propagation_delay            = lan
   dynamic_lnk_TG_user_defined_1               = 128
   dynamic_lnk_TG_user_defined_2               = 128
   dynamic_lnk_TG_user_defined_3               = 128
   comments                                    = ""
```

Figure 122. Token-Ring SNA DLC Profile

```
mode:
   prof_name                                   = "LU62APPB"
   mode_name                                   = "LU62APPB"
   max_sessions                                = 10
   min_conwinner_sessions                      = 5
   min_conloser_sessions                       = 0
   auto_activate_limit                         = 5
   max_adaptive_receive_pacing_window          = 16
   receive_pacing_window                       = 7
   max_ru_size                                 = 256
   min_ru_size                                 = 256
   class_of_service_name                       = "#CONNECT"
   comments                                    = ""
```

Figure 123. LU 6.2 Mode Profile

The local LU name must match the CICS/6000 region name and the host VTAM LU name for the CICS/6000 region. The remote LU name must be the same as the applid of the remote CICS.

A TPN, CICSTPN, is defined in the **local_tp** profile. This profile name must be added to the **TPNSNAProfile** attribute in the Transaction Definitions (TD) of your CICS/6000 region for all transactions that an SNA partner LU can access using transaction routing as well as the CICS intercommunication transactions, CRTE, CRSR, CPMI, CVMI, CSMI, CSM2, CSM3, and CSM5.

D.3 Encina PPC Gateway

Figure 124 shows the environment variables that must be added to the /etc/environment file after installation of the PPC Gateway.

```
ENCINA_CDS_ROOT=/.:/cics
ENCINA_GWY_SERVER=/.:/cics/ppc/gateway/GWYN
CICS_GWY_SERVER=/.:/cics/ppc/gateway/GWYN
ENCINA_AUTHN=1
```

Figure 124. Environment Variables Needed by the PPC Gateway

The value of ENCINA_CDS_ROOT must match the value of ENCINA_CDS_ROOT in your CICS/6000 region′s environment file.

D.4 DCE

A dedicated DCE principal must be created for the Encina PPC Gateway and account and keytab entries be added for this principal. Figure 125 shows how to create a DCE principal with the name *gwyn* and password *gwyn* using the DCE **rgy_edit** command. The DCE cell_admin password is represented by *<dce-passwd>* and should be replaced with your cell_admin password.

```
#rgy_edit
rgy_edit> domain principal
rgy_edit> add gwyn
rgy_edit> domain account
rgy_edit> add gwyn -g cics_admin -o none -pw gwyn -mp <dce-passwd>
rgy_edit> ktadd -p gwyn -pw gwyn
rgy_edit> exit
```

Figure 125. Creating a DCE Principal for PPC Gateway

Ensure that this principal has sufficient DCE acl rights to the following directories in the CDS:

- /.:/cics/*cics_region*
- /.:/cics/trpc
- /.:/cics/ppc
- /.:/cics/ppc/gateway.

This is achieved by using the command shown in Figure 126 on page 118.

```
acl_edit directory -m user :principal: rwdtcia
```

Figure 126. Granting DCE Permissions

The principal must also be given acl rights to the CICS/6000 region. Use the command shown in Figure 127.

```
acl_edit /.:/cics/cicsregion -io
-m user :principal: rwdtc
```

Figure 127. Granting Access Rights to CICS Region

Appendix E. Overcoming Full Function BMS Restrictions

The application we migrated to test scenarios 1 and 2 uses only minimum BMS functions, with no BMS restrictions. We include this appendix for those users who have full function BMS programs to migrate to the AIX environment.

Herein you can find a running example of how to modify your applications that use floating maps and/or logical message paging functions. We use the sample CICS paging transaction, REPT, as a base to provide you with a skeleton application that simulates the restricted full BMS functions in CICS/6000.

E.1 Sample DFH0CREP Program

The low balance report sample program, DFH0CREP, produces a report that lists all entries in the sample data set, FILEA, for which the amount is less than or equal to $50.00, using page-building techniques and illustrating the terminal paging facilities of BMS.

The transaction is invoked by entering the REPT transaction identifier on a clear screen. The program does a sequential scan through the file, selecting each entry that fulfills the search criteria. The pages are built from four maps that comprise the DFH0CGD mapset, using the PAGING option, so that the data is not displayed immediately but stored in temporary storage for later retrieval. The HEADING map is inserted at the top of each page. The LINE detail map is written repeatedly until the OVERFLOW condition occurs; that is, when there is no more free area space left in the screen. The FOOTING map is then written at the bottom of the page and the processing continues until the file scan is completed. Then the FINAL message map is written at the bottom of the last detail line, and the transaction ends (see Figure 128 for the distribution of the floating maps on the screen). You then have to clear the screen and enter the SIT-defined paging commands to retrieve the information.

In Figure 129 on page 120 you can find the source of the BMS mapset, and in Figure 131 on page 122, an annotated listing of the program.

```
┌─────────────────────────────────────────────────────┐
│  +----------------------------------------------+   │
│  |           H E A D I N G         PAGEN___ |   │
│  |        3 lines - 40 columns                  |   │
│  +----------------------------------------------+ -    - ---------
│               .................                          F    A
│               .................
│  +----------------------------------------------+       R    R
│  | LINE    1 line  - 40 columns                 |              (19 lines)
│  +----------------------------------------------+       E    E
│               .................
│               .................                          E    A
│  +----------------------------------------------+ -    - ---------
│  |           F O O T I N G                      |   │
│  |        2 lines - 40 columns                  |   │
│  +----------------------------------------------+   │
└─────────────────────────────────────────────────────┘
```

Figure 128. Floating Maps Distribution

E.2 Converting Maps from Floating to Fixed

As floating maps are not supported by CICS/6000, you will have to convert your set of floating maps to a full screen fixed map. Figure 129 shows the sample mapset DFH0CGD, and Figure 130 on page 121 shows the full screen fixed map.

```
         TITLE 'FILEA - MAPSET FOR LOW BALANCE REPORT - COBOL'
***********************************************************************
*                                                                     *
* MODULE NAME = DFHOCMD                                               *
*                                                                     *
* DESCRIPTIVE NAME = Report Map for Sample Application                *
*                                                                     *
*        5685-083                                                     *
*        COPYRIGHT = NONE                                             *
*                                                                     *
* STATUS = 3.3.0                                                      *
*                                                                     *
*---------------------------------------------------------------------*
*                                                                     *
* CHANGE ACTIVITY :                                                   *
* $SEG(DFHOCMD),COMP(SAMPLES),PROD(CICS/ESA):                         *
*                                                                     *
*     PN= REASON REL YYMMDD HDXIII : REMARKS                          *
*    $P0= .      320 900320        : Created.                         *
*    $P1= M90474 330 910807 HDBWSH : Prologue fixed.                  *
*                                                                     *
***********************************************************************
DFHOCGD  DFHMSD TYPE=&SYSPARM,MODE=OUT,CTRL=(FREEKB,FRSET),            *
               LANG=COBOL,STORAGE=AUTO,EXTATT=MAPONLY,COLOR=BLUE
*
LINE     DFHMDI SIZE=(1,40),COLOR=GREEN                     3
NUMBER   DFHMDF POS=(1,1),LENGTH=6
NAME     DFHMDF POS=(1,9),LENGTH=20
AMOUNT   DFHMDF POS=(1,30),LENGTH=8
*
HEADING  DFHMDI SIZE=(3,40),HEADER=YES                      1
         DFHMDF POS=(1,5),LENGTH=18,INITIAL='LOW BALANCE REPORT',      *
               HILIGHT=UNDERLINE
         DFHMDF POS=(1,24),LENGTH=1,ATTRB=PROT
         DFHMDF POS=(1,30),LENGTH=4,INITIAL='PAGE'
PAGEN    DFHMDF POS=(1,35),LENGTH=3
         DFHMDF POS=(3,1),LENGTH=6,INITIAL='NUMBER'
         DFHMDF POS=(3,17),LENGTH=4,INITIAL='NAME'
         DFHMDF POS=(3,32),LENGTH=6,INITIAL='AMOUNT'
*
FOOTING  DFHMDI SIZE=(2,40),TRAILER=YES,JUSTIFY=LAST        2
         DFHMDF POS=(2,1),LENGTH=38,                                   *
               INITIAL='PRESS CLEAR AND TYPE P/N TO SEE PAGE N'
*
FINAL    DFHMDI SIZE=(2,40),TRAILER=YES,JUSTIFY=LAST        4
         DFHMDF POS=(2,10),LENGTH=14,INITIAL='END OF REPORT.'
*
         DFHMSD TYPE=FINAL
         END
```

Figure 129. Sample DFH0CMD Mapset. You have a header floating map named HEADING(1); a trailer floating map named FOOTING(2); and two one-line floating maps: LINE(3), used to hold the detail lines, and FINAL(4), the final line of the report.

```
DFHOCGD DFHMSD TYPE=MAP,MODE=INOUT,CTRL=(FREEKB,FRSET),                X
               LANG=COBOL,STORAGE=AUTO,TERM=3270-2,COLOR=BLUE
*
DFHOCGD DFHMDI SIZE=(24,80),LINE=1,COLUMN=1                              1 5
*
HEADING DFHMDF POS=(1,5),LENGTH=18,INITIAL=′LOW BALANCE REPORT′,       X 2
               ATTRB=PROT
        DFHMDF POS=(1,24),LENGTH=1,ATTRB=PROT
        DFHMDF POS=(1,30),LENGTH=4,INITIAL=′PAGE′,ATTRB=PROT
PAGEN   DFHMDF POS=(1,33),LENGTH=3,ATTRB=(UNPROT,BRT,IC,FSET)
        DFHMDF POS=(1,37),LENGTH=1,ATTRB=ASKIP
        DFHMDF POS=(3,1),LENGTH=6,INITIAL=′NUMBER′,ATTRB=PROT
        DFHMDF POS=(3,17),LENGTH=4,INITIAL=′NAME′,ATTRB=PROT
        DFHMDF POS=(3,32),LENGTH=6,INITIAL=′AMOUNT′,ATTRB=PROT
*
LINE    DFHMDF POS=(4,1),LENGTH=79,OCCURS=19,ATTRB=PROT,COLOR=GREEN      4 6
*
FOOTING DFHMDF POS=(23,1),LENGTH=38,ATTRB=PROT,COLOR=RED,              X 3
               INITIAL=′PF3  PF7   PF8   PF10   PF11   PF12   ′
        DFHMDF POS=(24,1),LENGTH=38,ATTRB=PROT,COLOR=BLUE,             X
               INITIAL=′END  PREV  NEXT  LAST  RESHOW  GOTO   ′
        DFHMSD TYPE=FINAL
         END
```

Figure 130. Full Screen Fixed Mapset

Instead of several DFHMDI macros, you now only have one macro(1); that is, only one map, named DFH0CGD.

The fields of the former HEADING map are placed in fixed positions beginning at the upper left-hand corner of the screen(2), in lines 1 to 3. As the FOOTING map had two lines, the corresponding fields are positioned in lines 23 and 24(3).

The 19 lines of free area in the floating mapset are replaced by a LINE field with multiple occurrences(4). As CICS/6000 does not properly manage the screen position of the multiple fields when the map is not defined as 24 rows by 80 columns, we had to change the screen definition to SIZE=(24,80)(5), and hence the LINE definition from LENGTH=39 to LENGTH=79(6).

E.3 Program Modifications

The ACCUM option in SEND MAP statements tells BMS to accumulate pages of output data and send each page when it is complete. If PAGING is specified, BMS accumulates the pages in temporary storage, as device-dependent data streams, from where an operator can retrieve the stored pages by using the terminal operator paging transaction. This transaction is automatically initiated when a SEND PAGE command is encountered in the main transaction and starts when the main transaction RETURNs. The TS queue generated can be deleted by the main transaction by means of a PURGE MESSAGE command.

As CICS/6000 does not provide the paging function, we had to modify the program to emulate the TS queue buildup and code a new transaction PAGE (see E.5, “PAGEPGM Paging Program” on page 132) to emulate the CICS-supplied paging commands. Figure 131 on page 122 shows the original DFH0CREP program and the modifications we made. Figure 132 on page 129, shows the listing of the modified DFH0CREP program.

Notes

1. As the symbolic map name is included in the paging program, you have to code a different paging program for each full function BMS program you want to migrate.
2. To maintain consistent information in the temporary storage queue throughout main transaction invocations, you have to define the temporary storage queue as recoverable.

1

```
*****************************************************************
*                                                               *
* MODULE NAME = DFHOCREP                                        *
*                                                               *
* DESCRIPTIVE NAME = Low Balance Inquiry for Sample Application *
*                                                               *
*  5685-083                                                     *
*  COPYRIGHT = NONE                                             *
*                                                               *
* STATUS = 3.3.0                                                *
*                                                               *
*---------------------------------------------------------------*
*                                                               *
* CHANGE ACTIVITY :                                             *
* $SEG(DFHOCREP),COMP(SAMPLES),PROD(CICS/ESA):                  *
*                                                               *
*     PN= REASON REL YYMMDD HDXIII : REMARKS                    *
*    $P0= .      320 900320        : CREATED.                   *
*    $P1= M90474 330 910807 HDBWSH : Prologue fixed.            *
*                                                               *
*****************************************************************
 IDENTIFICATION DIVISION.
 PROGRAM-ID. FILECREP.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77  LOWLIM       PIC X(8)  VALUE '$0050.00'.
 77  KEYNUM       PIC 9(6)  VALUE 0.
*
* THE INPUT AREA FOR KEYED DTA AND THE MAXIMUM LENGHT OF KEYED
* DATA FOLLOW.
* IN PRACTICE, THE OPERATOR WILL ONLY PRESS ENTER.
*
 77  TERMDATA     PIC X(1).
 77  TERMLENG     PIC S9(4) COMP.
 77  PAGEN        PIC 9(3)  VALUE 1.
 77  OPINSTR      PIC X(52) VALUE 'PRESS THE ENTER KEY AND FOLLOW
-                                 'WITH PAGING COMMANDS.'.
 77  RESPONSE     PIC S9(8).
```

Figure 131 (Part 1 of 10). Sample DFH0CREP Program

As we changed the detail LINE map from a floating one-line map to a field array (OCCURS=19), we had to define an index variable for this array and a comparison field for AMOUNT:

```
000010 77  I            PIC S9(4) COMP VALUE 0.
000020 77  AMOUNTC      PIC X(8)  VALUE HIGH-VALUE.
```

2

```
        01  FILEA.   COPY DFHOCFIL.
       *
       * FILEA record expansion
       *
       *  02  FILEREC.
       *    03  STAT           PIC X.
       *    03  NUMB           PIC X(6).
       *    03  NOME           PIC X(20).
       *    03  ADDRX          PIC X(20).
       *    03  PHONE          PIC X(8).
       *    03  DATEX          PIC X(8).
       *    03  AMOUNT         PIC X(8).
       *    03  COMMENT        PIC X(9).
       *
                     COPY DFHOCGD.
       *
       * cicsmap DFHOCGD.bms -- Thu Jul 14 12:49:44 PDT 1994
       * DFHOCGD DFHMSD MODE=INOUT,STORAGE=AUTO
       *
       *01  DFHOCGDI.
       *    02   FILLER    PIC X(12).
       *    02   ATITLEL   PIC S9(4) COMP.
       *    02   ATITLEF   PIC X.
       *    02   ATITLEI   PIC X(1).
       *    02   PAGENL    PIC S9(4) COMP.
       *    02   PAGENF    PIC X.
       *    02   PAGENI    PIC X(3).
       *    02   LINED OCCURS 19 TIMES.
       *      03 LINEL     PIC S9(4) COMP.
       *      03 LINEF     PIC X.
       *      03 LINEI     PIC X(79).
       *01  DFHOCGDO REDEFINES DFHOCGDI.
       *    02   FILLER    PIC X(12).
       *    02   FILLER    PIC S9(4) COMP.
       *    02   ATITLEA   PIC X.
       *    02   ATITLEO   PIC X(1).
       *    02   FILLER    PIC S9(4) COMP.
       *    02   PAGENA    PIC X.
       *    02   PAGENO    PIC X(3).
       *    02   DFHMS01 OCCURS 19 TIMES.
       *      03 FILLER    PIC S9(4) COMP.
       *      03 LINEA     PIC X.
       *      03 LINEO     PIC X(79).
```

Figure 131 (Part 2 of 10). Sample DFH0CREP Program

As we changed the LINE map to a ′PIC X(79)′ character string, we had to remap the former NUMBER, NAME, and AMOUNT map fields to the new LINE array, by means of the following redefinition:

```
000030 02  FILLER REDEFINES DFHMS01 OCCURS 19 TIMES.
000040   03 FILLER      PIC S9(4) COMP.
000050   03 FILLER      PIC X.
000060   03 NUMBERO     PIC X(6).
000070   03 FILLER      PIC X(2).
000080   03 NAMEO       PIC X(20).
000090   03 FILLER      PIC X(1).
000100   03 AMOUNTO     PIC X(8).
000110   03 FILLER      PIC X(42).
```

We used as temporary storage queue name the concatenation of the CICS terminal name, TERMID, with the paging transaction name (′PAGE′). Therefore, we had to do the appropriate variable definition:

```
000120 01  TSNAME.
000130   03 TRANID      PIC X(4) VALUE ′PAGE′.
000140   03 TERMNID     PIC X(4).
```

3

```
        PROCEDURE DIVISION.
       *
       *    THE ERROR EXITS ARE SET UP. NOTE THAT THE ″ERROR″ CONDITION
       *    IS SPECIFIED TO DETECT ANY UNEXPECTED ERRORS, ALL OTHER
       *    ERRORS ARE DETECTED EXPLICITLY BY USING THE ″RESP″ OPTION
       *    ON CICS COMMANDS. NOTE THAT AN EXIT MUST BE SPECIFIED FOR
       *    THE ″OVERFLOW″ CONDITION OTHERWISE CICS WILL NOT ALLOW THIS
       *    RESPONSE TO A ″SEND MAP″ COMMAND, EVEN THOUGH THIS RESPONSE
       *    IS DETECTED BY USE OF THE ″RESP″ OPTION.
       *
```

Figure 131 (Part 3 of 10). Sample DFH0CREP Program

We first loaded the CICS terminal name into the temporary storage name and then deleted any previous occurrence of the temporary storage queue:

```
000150     MOVE EIBTRMID TO TERMNID.
000160     EXEC CICS HANDLE CONDITION QIDERR(CONT) END-EXEC.
000170     EXEC CICS DELETEQ TS QUEUE(TSNAME) END-EXEC.
000180 CONT.
```

4

```
            EXEC CICS HANDLE CONDITION ERROR(ERRORS)
                             OVERFLOW(OFLOW) END-EXEC.
            MOVE LOW-VALUE TO PAGENA
       *
       *    A PAGE NUMBER OF 1 IS MOVED TO THE HEADING MAP.
       *
            MOVE PAGEN     TO PAGENO
       *
       *    THIS ″BMS″ COMMAND SETS UP THE HEADING IN THE PAGE BUILD
       *    OPERATION, ″BMS″ BUILDS THE PAGES IN TEMPORARY STORAGE.
       *
            EXEC CICS SEND MAP(′HEADING′) MAPSET(′DFHOCGD′) ACCUM
                 PAGING ERASE END-EXEC
```

Figure 131 (Part 4 of 10). Sample DFH0CREP Program

As the heading is a part of the modified map, we had to comment out the last call.

Before starting the browse, we initialized the index counter:

```
 000190     MOVE 1 TO I.
```

5

```
      *
      *    THE "STARTBR" COMMAND SETS UP THE FILE BROWSE TO BEGIN AT
      *    THE FIRST RECORD WITH A KEY EQUAL TO OR GREATER THAN THE
      *    "RIDFLD", IN THIS CASE THE FIRST RECORD ON FILE.
      *
           EXEC CICS STARTBR FILE('FILEA') RIDFLD(KEYNUM) END-EXEC.
       REPEAT.
      *
      *    THIS COMMAND READS THE NEXT CUSTOMER RECORD FROM "FILEA".
      *    NOTE THE TESTING OF THE RESPONSE TO THE COMMAND.
      *
           EXEC CICS READNEXT INTO(FILEA) RIDFLD(KEYNUM)
                               RESP(RESPONSE) FILE('FILEA') END-EXEC
      *
      *    CHECK RESPONSES
      *
           IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO ENDFILE.
           IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
           MOVE AMOUNT    TO AMOUNTO
      *
      *    THE SEARCH CRITERION FOR CREATING THE REPORT IS THAT THE
      *    CUSTOMER HAS A BANK BALANCE WHICH IS $50 OR LESS.
      *
           IF AMOUNTO GREATER THAN LOWLIM GO TO REPEAT.
           MOVE LOW-VALUE TO LINEO
           MOVE AMOUNT    TO AMOUNTO
      *
      *    FIELDS ARE MOVED FROM THE SELECTED CUSTOMER RECORD TO THE
      *    MAP AREA FOR THE DETAIL LINE.
      *
           MOVE NUMB      TO NUMBERO
           MOVE NOME      TO NAMEO
```

Figure 131 (Part 5 of 10). Sample DFH0CREP Program

As we changed the AMOUNT map field to a redefinition on the array, we had to use a comparison variable instead and move the FILEA record fields to the redefined subscripted fields:

```
000200     MOVE AMOUNT    TO AMOUNTC.
000210     IF AMOUNTC GREATER THAN LOWLIM GO TO REPEAT.
000220     MOVE SPACES    TO LINEO(I).
000230     MOVE AMOUNT    TO AMOUNTO(I).
000240     MOVE NUMB      TO NUMBERO(I).
000250     MOVE NOME      TO NAMEO(I).
```

6

```
    *
    *    THE CUSTOMER DETAIL MAP IS SET UP FOR SUBSEQUENT PAGING.
    *
         EXEC CICS SEND MAP('LINE') MAPSET('DFH0CGD')
                   ACCUM PAGING RESP(RESPONSE) END-EXEC
    *
    *    CHECK RESPONSES
    *
         IF RESPONSE = DFHRESP(OVERFLOW) THEN GO TO OFLOW.
         IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
         GO TO REPEAT.
```

Figure 131 (Part 6 of 10). Sample DFH0CREP Program

Instead of ACCUMulating a detail LINE map, we had to increment the index counter and branch to OFLOW if the variable I got out of limits:

```
000260     ADD 1  TO I.
000270     IF I    >     19                THEN GO TO OFLOW.
         GO TO REPEAT.
```

7

```
     ENDFILE.
    *
    *    WHEN THE "ENDFILE" CONDITION IS RAISED, THE LAST MAP IS SENT
    *    TO "BMS".
    *
         EXEC CICS SEND MAP('FINAL') MAPSET('DFH0CGD')
                   MAPONLY ACCUM PAGING END-EXEC
    *
    *    THE "SEND PAGE" COMMAND MAKES ALL THE PAGES OF THE REPORT
    *    AVAILABLE FOR PAGING AT THE TERMINAL, WHEN THE CURRENT
    *    TRANSACTION TERMINATES.
    *
         EXEC CICS SEND PAGE END-EXEC
```

Figure 131 (Part 7 of 10). Sample DFH0CREP Program

Instead of sending the FINAL floating map, we moved the final message to the detail array element, initialized the remainder of the array with binary zeroes, and wrote the last map to temporary storage:

```
     MOVE SPACES TO LINEO(I).
     MOVE 'END OF REPORT.' TO NAMEO(I).
     ADD 1  TO I.
 ENDFILE-LOOP.
     IF I    >     19               THEN GO TO STORE-MAP.
     MOVE LOW-VALUE TO LINEO(I).
     ADD  1         TO I.
     GO TO ENDFILE-LOOP.
 STORE-MAP.
     EXEC CICS WRITEQ TS QUEUE(TSNAME) FROM(DFHOCGDO)
                         END-EXEC.
```

8

```
 *
 *    A MESSAGE IS SENT TO THE TERMINAL. THIS MESSAGE WILL BE
 *    DISPLAYED BEFORE THE PAGES OF THE LOW BALANCE REPORT.
 *
      EXEC CICS SEND TEXT FROM(OPINSTR) LENGTH(52)
                ERASE END-EXEC
 *
 *    THE FILE BROWSE OPERATION IS TERMINATED.
 *
      EXEC CICS ENDBR FILE('FILEA') END-EXEC
 *
 *    A RECEIVE COMMAND IS ISSUED TO GIVE THE TERMINAL OPERATOR
 *    A CHANCE TO READ THE PROMPTING MESSAGE.
 *
 *    THE TRANSACTION WILL TERMINATE WHEN THE OPERATOR PRESSES THE
 *    ENTER KEY.
 *
 *    PAGING COMMANDS CAN THEN BE ISSUED.
 *
 *    NO HARM IS DONE IF THE OPERATOR TYPES IN DATA BEFORE
 *    PRESSING THE ENTER KEY.
 *
 *    THE "RECEIVE MAP" COMMAND READS FROM THE TERMINAL AND ALLOWS
 *    THE TERMINAL OPERATOR TO READ THE PROMPTING MESSAGE BEFORE
 *    THE FIRST PAGE OF THE REPORT IS DISPLAYED.
 *
      EXEC CICS RECEIVE INTO(TERMDATA) LENGTH(TERMLENG)
                RESP(RESPONSE) END-EXEC.
 *
 *    CHECK RESPONSE - IGNORE LENGERR
 *
      IF NOT (RESPONSE = DFHRESP(NORMAL)
              OR RESPONSE = DFHRESP(EOC)
              OR RESPONSE = DFHRESP(LENGERR))
         THEN GO TO ERRORS.
  ENDTASK.
 *
 *    THE PROGRAM ENDS, THE FIRST PAGE OF THE REPORT WILL NOW BE
 *    DISPLAYED.
 *
      EXEC CICS RETURN END-EXEC.
      GOBACK.
```

Figure 131 (Part 8 of 10). Sample DFH0CREP Program

As the way of starting the paging transaction changes from the automatic invocation in full function BMS to a RETURN TRANSID in CICS/6000, which mandates an ENTER to start the new transaction, we had to omit the RECEIVE command and the subsequent return code check, to avoid a redundant ENTER. We also had to change the RETURN statement to a RETURN TRANSID:

```
000380     EXEC CICS RETURN TRANSID(TRANID) END-EXEC.
```

9

```
     ERRORS.
    *
    *    IF THE "ERROR" CONDITION OCCURS ON A CICS COMMAND, THIS
    *    ROUTINE GAINS CONTROL. HANDLING OF THE "ERROR" CONDITION IS
    *    SUPRESSED, ANY DATA SENT TO "BMS" IS PURGED AND THE PROGRAM
    *    TERMINATES ABNORMALLY WITH A TRANSACTION DUMP.
    *
         EXEC CICS PURGE MESSAGE NOHANDLE END-EXEC
         EXEC CICS ABEND ABCODE('ERRS') NOHANDLE END-EXEC.
```

Figure 131 (Part 9 of 10). Sample DFH0CREP Program

Instead of PURGEing the messages stored, we deleted the temporary storage queue and reset the ERROR condition to avoid an abend loop:

```
000390     EXEC CICS DELETEQ TS QUEUE(TSNAME) END-EXEC.
000400     EXEC CICS HANDLE CONDITION ERROR END-EXEC.
```

10

```
      OFLOW.
    *
    *    IF THE PROGRAM DETECTS THE "OVERFLOW" CONDITION WHEN A DETAIL
    *    LINE IS SENT TO "BMS", THIS ROUTINE GAINS CONTROL.
    *    THIS ROUTINE COMPLETES THE CURRENT PAGE AND STARTS THE NEXT
    *    ONE. THIS "BMS" COMMAND SETS UP THE FOOTING FOR THE CURRENT
    *    PAGE.
    *
         EXEC CICS SEND MAP('FOOTING') MAPSET('DFH0CGD')
                           MAPONLY ACCUM PAGING END-EXEC
         ADD 1      TO PAGEN
         MOVE PAGEN TO PAGENO
    *
    *    THIS "BMS" COMMAND SETS UP THE HEADING FOR THE NEXT PAGE.
    *
         EXEC CICS SEND MAP('HEADING') MAPSET('DFH0CGD')
                   ACCUM PAGING ERASE END-EXEC.
    *
    *    THIS "BMS" COMMAND RESENDS THE DETAIL LINE WHICH CAUSED THE
    *    "OVERFLOW" CONDITION.
    *
         EXEC CICS SEND MAP('LINE') MAPSET('DFH0CGD')
                   ACCUM PAGING END-EXEC
         GO TO REPEAT.
```

Figure 131 (Part 10 of 10). Sample DFH0CREP Program

As the FOOTING and HEADING maps are now part of the modified fixed map, we had to omit the sendings, write the map output stream, DFH0CGDO, to temporary storage instead, and reinitialize the array index variable:

```
000410     EXEC CICS WRITEQ TS QUEUE(TSNAME) FROM(DFHOCGDO)
000411                         END-EXEC.
000420     MOVE 1     TO I.
```

E.4 Modified Sample DFH0CREP Program

Figure 132 shows the modified source, with the original comments stripped out. The added or modified lines have COBOL sequence numbers on them, and the suppressed original statements are commented and have an 'O' in column 8.

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID. FILECREP.
       ENVIRONMENT DIVISION.
       DATA DIVISION.
       WORKING-STORAGE SECTION.
       77  LOWLIM       PIC X(8)  VALUE '$0050.00'.
       77  KEYNUM       PIC 9(6)  VALUE 0.
       77  TERMDATA     PIC X(1).
       77  TERMLENG     PIC S9(4) COMP.
       77  PAGEN        PIC 9(3)  VALUE 1.
       77  OPINSTR      PIC X(52) VALUE 'PRESS THE ENTER KEY AND FOLLOW
      -                              'WITH PAGING COMMANDS.'.
       77  RESPONSE     PIC S9(8).
000010 77  I            PIC S9(4) COMP VALUE 0.
000020 77  AMOUNTC      PIC X(8)  VALUE HIGH-VALUE.
      *
       01  FILEA.   COPY DFH0CFIL.
      *
                    COPY DFH0CGD.
      *
000030 02  FILLER REDEFINES DFHMS01 OCCURS 19 TIMES.
000040   03 FILLER      PIC S9(4) COMP.
000050   03 FILLER      PIC X.
000060   03 NUMBERO     PIC X(6).
000070   03 FILLER      PIC X(2).
000080   03 NAMEO       PIC X(20).
000090   03 FILLER      PIC X(1).
000100   03 AMOUNTO     PIC X(8).
000110   03 FILLER      PIC X(42).
      *
000120 01  TSNAME.
000130   03 TRANID      PIC X(4) VALUE 'PAGE'.
000140   03 TERMNID     PIC X(4).
```

Figure 132 (Part 1 of 4). Modified DFH0CREP Program

```
       PROCEDURE DIVISION.
000150     MOVE EIBTRMID TO TERMNID.
000160     EXEC CICS HANDLE CONDITION QIDERR(CONT) END-EXEC.
000170     EXEC CICS DELETEQ TS QUEUE(TSNAME) END-EXEC.
000180 CONT.
      *
           EXEC CICS HANDLE CONDITION ERROR(ERRORS)
                            OVERFLOW(OFLOW) END-EXEC.
           MOVE LOW-VALUE TO PAGENA.
           MOVE PAGEN     TO PAGENO.
      *
      *O   EXEC CICS SEND MAP(’HEADING’) MAPSET(’DFH0CGD’) ACCUM
      *O        PAGING ERASE END-EXEC.
      *
000190     MOVE 1 TO I.
      *
           EXEC CICS STARTBR FILE(’FILEA’) RIDFLD(KEYNUM) END-EXEC.
       REPEAT.
           EXEC CICS READNEXT INTO(FILEA) RIDFLD(KEYNUM)
                              RESP(RESPONSE) FILE(’FILEA’) END-EXEC.
           IF RESPONSE = DFHRESP(ENDFILE) THEN GO TO ENDFILE.
           IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
      *
000200     MOVE AMOUNT    TO AMOUNTC.
000210     IF AMOUNTC GREATER THAN LOWLIM GO TO REPEAT.
000220     MOVE SPACES    TO LINEO(I).
000230     MOVE AMOUNT    TO AMOUNTO(I).
000240     MOVE NUMB      TO NUMBERO(I).
000250     MOVE NOME      TO NAMEO(I).
      *
      *O   EXEC CICS SEND MAP(’LINE’) MAPSET(’DFH0CGD’)
      *O             ACCUM PAGING RESP(RESPONSE) END-EXEC.
      *O   IF RESPONSE = DFHRESP(OVERFLOW) THEN GO TO OFLOW.
      *O   IF RESPONSE NOT = DFHRESP(NORMAL) THEN GO TO ERRORS.
      *
000260     ADD 1  TO I.
000270     IF I    >     19               THEN GO TO OFLOW.
      *
           GO TO REPEAT.
```

Figure 132 (Part 2 of 4). Modified DFH0CREP Program

```
       ENDFILE.
      *
      *O   EXEC CICS SEND MAP('FINAL') MAPSET('DFHOCGD')
      *O             MAPONLY ACCUM PAGING END-EXEC
      *O   EXEC CICS SEND PAGE END-EXEC
      *
000280     MOVE SPACES TO LINEO(I).
000290     MOVE 'END OF REPORT.' TO NAMEO(I).
000300     ADD 1  TO I.
000310 ENDFILE-LOOP.
000320     IF I    >     19                THEN GO TO STORE-MAP.
000330     MOVE LOW-VALUE TO LINEO(I).
000340     ADD  1         TO I.
000350     GO TO ENDFILE-LOOP.
000360 STORE-MAP.
000370     EXEC CICS WRITEQ TS QUEUE(TSNAME) FROM(DFHOCGDO)
000371                          END-EXEC.
      *
           EXEC CICS SEND TEXT FROM(OPINSTR) LENGTH(52)
                     FREEKB ERASE END-EXEC.
           EXEC CICS ENDBR FILE('FILEA') END-EXEC.
      *
      *O   EXEC CICS RECEIVE INTO(TERMDATA) LENGTH(TERMLENG)
      *O             RESP(RESPONSE) END-EXEC.
      *O   IF NOT (RESPONSE = DFHRESP(NORMAL)
      *O           OR RESPONSE = DFHRESP(EOC)
      *O           OR RESPONSE = DFHRESP(LENGERR))
      *O      THEN GO TO ERRORS.
      *
       ENDTASK.
      *
      *O   EXEC CICS RETURN END-EXEC.
      *
000380     EXEC CICS RETURN TRANSID(TRANID) END-EXEC.
      *
           GOBACK.
      *
```

Figure 132 (Part 3 of 4). Modified DFH0CREP Program

```
        ERRORS.
       *
       *O   EXEC CICS PURGE MESSAGE NOHANDLE END-EXEC
       *
 000390     EXEC CICS DELETEQ TS QUEUE(TSNAME) END-EXEC.
 000400     EXEC CICS HANDLE CONDITION ERROR END-EXEC.
       *
            EXEC CICS ABEND ABCODE('ERRS') NOHANDLE END-EXEC.
        OFLOW.
       *
       *O   EXEC CICS SEND MAP('FOOTING') MAPSET('DFH0CGD')
       *O                     MAPONLY ACCUM PAGING END-EXEC.
       *
 000410     EXEC CICS WRITEQ TS QUEUE(TSNAME) FROM(DFH0CGDO)
 000411                        END-EXEC.
 000420     MOVE 1     TO I.
       *
            ADD 1      TO PAGEN.
            MOVE PAGEN TO PAGENO.
       *
       *O   EXEC CICS SEND MAP('HEADING') MAPSET('DFH0CGD')
       *O             ACCUM PAGING ERASE END-EXEC.
       *O   EXEC CICS SEND MAP('LINE') MAPSET('DFH0CGD')
       *O             ACCUM PAGING END-EXEC.
       *
            GO TO REPEAT.
```

Figure 132 (Part 4 of 4). Modified DFH0CREP Program

E.5 PAGEPGM Paging Program

The paging program, PAGEPGM, reads a particular record from the temporary storage queue generated by the main transaction, sends the corresponding map to the screen, and ends invoking itself in a pseudo-conversational way.

If it is the first invocation, the program reads and displays the first map stored; in subsequent invocations, it will display the maps requested as per the program function keys (PFKs) pressed. This program uses the PAGEN field to know which map record was read before. This is why we had to change the map mode from MODE=OUT to MODE=INOUT. You cannot do a RECEIVE MAP in a MODE=OUT map. You can choose a different way to handle the record index between invocations; for example, use the COMMAREA to communicate the previous item number.

The PAGEPGM program implements the following functions:

- Scroll forward
- Scroll backward
- Position in a selected page
- Position in the last page
- Reshow the current page
- Quit scrolling.

When the program receives the END command, it deletes the temporary storage queue and ends the transaction.

You can use this program as a skeleton and modify it to suit your needs. Figure 133 on page 134 shows the program listing, and Figure 134 on page 136, the corresponding mapset.

As the symbolic map name is included in the PAGEPGM program, you have to code a different paging program for each full function BMS program you want to migrate.

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID. PAGEPGM.
 ENVIRONMENT DIVISION.
 DATA DIVISION.
 WORKING-STORAGE SECTION.
 77  I            PIC 9(4) COMP VALUE 1.
 77  ITNO         PIC 9(4) COMP VALUE 1.
 77  QITEMS       PIC 9(4) COMP VALUE 0.
*
 77  PAGEN        PIC 9(3) VALUE 1.
*
 01  TSNAME.
   03 TERMNID     PIC X(4).
   03 TRANID      PIC X(4) VALUE 'PAGE'.
*
             COPY DFHAID.
*
             COPY PAGEMAP.
*
             COPY DFHOCGD.
*
********************
 PROCEDURE DIVISION.
********************
     MOVE EIBTRMID TO TERMNID.
     MOVE 1        TO PAGEN.
     EXEC CICS HANDLE AID CLEAR(TERMIN) END-EXEC.
     EXEC CICS HANDLE CONDITION ERROR(ERRORS) MAPFAIL(READTSQ)
            QIDERR(NOMAPQ) ITEMERR(OUTBNDS) END-EXEC.
*
     EXEC CICS RECEIVE MAP('DFHOCGD') MAPSET('DFHOCGDM')
               END-EXEC.
     MOVE PAGENI TO PAGEN.
     IF EIBAID = DFHPF3   THEN GO TO TERMIN.
     IF EIBAID = DFHPF7   THEN GO TO PGUP.
     IF EIBAID = DFHPF8   THEN GO TO PGDN.
     IF EIBAID = DFHENTER THEN GO TO PGDN.
     IF EIBAID = DFHPF10  THEN GO TO LASTPG.
     IF EIBAID = DFHPF11  THEN GO TO RESHOW.
     IF EIBAID = DFHPF12  THEN GO TO GOTOPG.
     EXEC CICS SEND MAP('INVPFK') MAPSET('PAGEMAP')
               ERASE FREEKB MAPONLY END-EXEC.
     EXEC CICS RECEIVE MAP('INVPFK') MAPSET('PAGEMAP')
               END-EXEC.
*
 RESHOW.
     MOVE PAGEN    TO ITNO.
     EXEC CICS READQ TS QUEUE(TSNAME) INTO(DFHOCGDO)
            NUMITEMS(QITEMS) ITEM(ITNO) END-EXEC.
*
* ATTRIBUTE '5' STANDS FOR PROTECTED MODIFIED AUTOSKIP PEN-SEL
*
     MOVE '5'      TO PAGENA.
     EXEC CICS SEND MAP('DFHOCGD') MAPSET('DFHOCGDM')
               ERASE FREEKB END-EXEC.
*
     EXEC CICS RETURN TRANSID(TRANID) END-EXEC.
*
     GOBACK.
*
```

Figure 133 (Part 1 of 2). PAGEPGM Paging Program

```
 READTSQ.
     MOVE 1        TO PAGEN.
     GO TO RESHOW.
*
 PGUP.
     SUBTRACT 1  FROM PAGEN.
     GO TO RESHOW.
*
 PGDN.
     ADD  1        TO PAGEN.
     GO TO RESHOW.
*
 LASTPG.
     MOVE 1        TO PAGEN.
     MOVE 1        TO ITNO.
     EXEC CICS READQ TS QUEUE(TSNAME) INTO(DFHOCGDO)
            NUMITEMS(QITEMS) ITEM(ITNO) END-EXEC.
     MOVE QITEMS   TO PAGEN.
     GO TO RESHOW.
*
 TERMIN.
     EXEC CICS SEND MAP('TERMIN') MAPSET('PAGEMAP')
               ERASE FREEKB MAPONLY END-EXEC.
     EXEC CICS DELETEQ TS QUEUE(TSNAME) END-EXEC.
     EXEC CICS RETURN END-EXEC.
*
     GOBACK.
*
 OUTBNDS.
     MOVE PAGEN    TO PGINVO.
     EXEC CICS SEND MAP('OUTBNDS') MAPSET('PAGEMAP')
               ERASE FREEKB END-EXEC.
     EXEC CICS RECEIVE MAP('OUTBNDS') MAPSET('PAGEMAP')
               END-EXEC.
*
 GOTOPG.
     MOVE PAGEN    TO PGNO.
     EXEC CICS SEND MAP('GOTOPG') MAPSET('PAGEMAP')
               ERASE FREEKB END-EXEC.
     EXEC CICS RECEIVE MAP('GOTOPG') MAPSET('PAGEMAP')
               END-EXEC.
     MOVE PGNI     TO PAGEN.
     GO TO RESHOW.
*
 ERRORS.
     EXEC CICS DELETEQ TS QUEUE(TSNAME) END-EXEC.
     EXEC CICS ABEND ABCODE('ERRS') NOHANDLE END-EXEC.
*
 NOMAPQ.
     EXEC CICS SEND MAP('NOMAPQ') MAPSET('PAGEMAP')
               ERASE FREEKB MAPONLY END-EXEC.
     EXEC CICS RETURN END-EXEC.
*
     GOBACK.
```

Figure 133 (Part 2 of 2). PAGEPGM Paging Program

```
PAGEMAP  DFHMSD TYPE=&SYSPARM,MODE=INOUT,CTRL=(FREEKB,FRSET,L40),      *
               LANG=COBOL,STORAGE=AUTO,COLOR=BLUE,TIOAPFX=YES
*
INVPFK   DFHMDI SIZE=(09,40),LINE=1,COLUMN=1
         DFHMDF POS=(1,1),LENGTH=39,                                   *
               INITIAL='INVALID OPTION SELECTED. HIT <ENTER>   '
         DFHMDF POS=(2,1),LENGTH=39,                                   *
               INITIAL='TO CONTINUE OR <CLEAR>  TO END.        '
         DFHMDF POS=(4,1),LENGTH=39,                                   *
               INITIAL='PF3  / CLEAR          = END            '
         DFHMDF POS=(5,1),LENGTH=39,                                   *
               INITIAL='PF7                   = PREV PAGE      '
         DFHMDF POS=(6,1),LENGTH=39,                                   *
               INITIAL='PF8  / ENTER          = NEXT PAGE      '
         DFHMDF POS=(7,1),LENGTH=39,                                   *
               INITIAL='PF10                  = LAST PAGE      '
         DFHMDF POS=(8,1),LENGTH=39,                                   *
               INITIAL='PF11                  = RESHOW PAGE    '
         DFHMDF POS=(9,1),LENGTH=39,                                   *
               INITIAL='PF12                  = GO TO PAGE NNN '
*
TERMIN   DFHMDI SIZE=(01,40),LINE=1,COLUMN=1
         DFHMDF POS=(1,1),LENGTH=39,                                   *
               INITIAL='PAGE PROCESSING COMPLETED SUCCESSFULLY.'
*
GOTOPG   DFHMDI SIZE=(02,40),LINE=1,COLUMN=1
         DFHMDF POS=(1,1),LENGTH=39,ATTRB=PROT,                        *
               INITIAL='ENTER NUMBER OF PAGE OR <CLEAR> TO END.'
         DFHMDF POS=(2,5),LENGTH=12,ATTRB=PROT                         *
               INITIAL='GO TO PAGE :'
PGN      DFHMDF POS=(2,25),LENGTH=3,COLOR=GREEN,ATTRB=(UNPROT,IC,FSET)
*
OUTBNDS  DFHMDI SIZE=(03,40),LINE=1,COLUMN=1
         DFHMDF POS=(1,1),LENGTH=39,                                   *
               INITIAL='INVALID NUMBER OF PAGE. HIT <ENTER>    '
         DFHMDF POS=(2,1),LENGTH=39,                                   *
               INITIAL='TO CONTINUE OR <CLEAR>  TO END.        '
         DFHMDF POS=(3,5),LENGTH=12,                                   *
               INITIAL='      PAGE :'
PGINV    DFHMDF POS=(3,25),LENGTH=3,COLOR=RED
*
NOMAPQ   DFHMDI SIZE=(02,40),LINE=1,COLUMN=1
         DFHMDF POS=(1,1),LENGTH=39,                                   *
               INITIAL='PAGE REQUEST RECEIVED BUT THERE ARE NO '
         DFHMDF POS=(2,1),LENGTH=39,                                   *
               INITIAL='PAGES TO DISPLAY. PROCESSING TERMINATED'
*
         DFHMSD TYPE=FINAL
         END
```

Figure 134. Paging PAGESET Mapset

Appendix F. JCL Conversion

The concept of JCL as used on the mainframe does not exist in AIX. JCL that is used to execute applications on the mainframe must be converted to AIX shell scripts.

The application we migrated during this residency is completely interactive and online, with no JCL requirements. We include this appendix for those users that have to migrate MVS JCL to an AIX script format.

F.1 JCL Revision

Figure 135 on page 138 illustrates a typical batch MVS jobstream.

```
//DSNTIJXB JOB (acct,info),'program name',CLASS=A,            2  3
//             MSGCLASS=T,NOTIFY=USER1                         30 34
//*************************************************            1
//*                                               *
//*  DESCRIPTIVE NAME = SAMPLE JCL TO CONVERT     *
//*                                               *
//*************************************************
//JOBLIB  DD  DSN=MY.LOAD.LIBRARY,DISP=SHR                     23
//        DD  DSN=MY.SECOND.LOAD.LIBRARY,DISP=SHR              24
//*
//MYPROC  PROC WKSPC=5,                                        6
//             MEM=CTLMEMBR                                    11
//PROCSTP1 EXEC PGM=FIRSTPGM,
//             PARM='PARM1,PARM2'                              9
//RUNCTL   DD  DSN=MY.PARTIT.DATASET(&MEM),DISP=SHR            27
//MYFILE   DD  DSN=MY.VSAM.FILE,DISP=SHR                       25
//MYOUTPUT DD  DSN=&&TEMPOR,DISP=(NEW,PASS),UNIT=SYSDA,        28
//             SPACE=(CYL,(&WKSPC,&WKSPC)),VOL=SER=DISK01,     12
//             DCB=(LRECL=120,RECFM=FB,BLKSIZE=8040)
//SYSPRINT DD  SYSOUT=*                                        29
//SYSUDUMP DD  DUMMY
//*
//PROCSTP2 EXEC PGM=SECNDPGM,                                  4
//             COND=(4,LT,PROCSTP1)                            8
//MYINPUT  DD  DSN=&&TEMPOR,DISP=(OLD,DELETE)
//MYREPORT DD  SYSOUT=*
//MSTFLE   DD  DSN=ANY.CATLG.FILE,DISP=SHR                     26
//SYSUT1   DD  UNIT=SYSDA,SPACE=(1024,(50,50))
//NEWFILE  DD  DSN=MY.NEW.FILE,                                13 14
//             UNIT=SYSDA,                                     15
//             VOL=SER=DISK01,                                 16
//             DISP=(NEW,CATLG,DELETE),                        17
//             SPACE=(CYL,(20,5)),                             18
//             DCB=(LRECL=80,                                  19 20
//             RECFM=FB,                                       21
//             BLKSIZE=8000)                                   22
//MYPROC  PEND                                                 7
//*
//PH01S01 EXEC MYPROC,                                         5
//             WKSPC=10                                        10
//PROCSTP2.SYSIN    DD *                                       31
 data for my program lrecl=80
/*                                                             32
//*
//                                                             33
```

Figure 135. Sample Batch JCL

All JCL statements begin with '//' in columns 1-2, span until column 72 inclusive, and have to be in uppercase.

- Comment statements have an asterisk in column 3 (1).
- Continuation of previous statements have a blank in column 3 and the last nonblank character must be a comma (2).
- The JOB statement (3) identifies the batch job.
- The EXEC statement identifies the member to be executed: a program (PGM= 4), or a procedure (5).

- A procedure is delimited by the PROC statement(6) and the PEND statement(7). Procedures may be stored in-stream, as in the example, or in a partitioned procedure library.
- The execution of a step may be conditioned by the return code of a previous step (COND= 8). The COND= keyword has three positional parameters:
  1. The first parameter is the value to be compared with the return code of the step coded in the third parameter.
  2. The second parameter is the logical operation to be performed. Possible values are lesser-than (LT), lesser-or-equal (LE), not-lesser-than (NL), greater-than (GT), greater-or-equal (GE), not-greater-than (NG), equal (EQ), not-equal (NE).
  3. The return code of the step whose name is coded in the third parameter will be compared with the value coded in the first parameter.

  Then, COND=(4,LT,PROCSTP1) will be interpreted as follows: IF 4 is lesser than the return code of step PROCSTP1, THEN the current step (PROCSTP2) will **NOT** be executed.
- A program may receive run-time parameters through the PARM= keyword(9).
- A procedure may receive run-time (10) or preset (11) values for the replaceable parameters (&var 12).
- The data definition (DD) statement defines where and how the physical data is stored (13). The symbolic name of the data set is found in columns 3 through 10.
- The DSN= keyword identifies the actual name of the data set in the storage device (14).
- The type of storage device is defined through the UNIT= keyword (15). SYSDA is a symbolic name to address any type of disk. You may find specific types of devices (for example, 3480, 3490, 3380, 3390).
- The label of the physical data support is provided in the VOL=SER= keyword (16).
- The disposition of the data set is given in the DISP= keyword (17). It has three positional parameters:
  1. The first parameter indicates whether the file is nonexistent (NEW), already defined (OLD) with exclusive access intended, already defined with shared access intended (SHR), or to be appended (MOD).
  2. The second parameter states how the job will treat the data set upon normal termination of the step. It can:
     a. KEEP the data set in the storage device
     b. Keep and catalog the data set (CATLG)
     c. Keep the data set but uncatalog it (UNCATLG)
     d. DELETE the data set (and uncatalog it if it is already cataloged).
  3. The third parameter states how the job will treat the data set upon abnormal termination of the step. It has the same value options as the second parameter.
- The SPACE= keyword (18) indicates the amount of storage the data set will allocate. It has several positional parameters; the first four are:
  1. The first parameter indicates the unit of allocation: cylinders (CYL), tracks (TRK), or blocks (a number indicating block length).
  2. The second parameter indicates the amount of these units of allocation. It has three positional subparameters: the primary allocation, the secondary allocation, and the quantity of directory blocks, if it is a partitioned data set.
  3. The third parameter, if present, indicates that the unused space will be freed when closing the data set (RLSE).

4. The fourth parameter, if present, indicates that the allocation will be made in contiguous units (CONTIG).

- The data control block (DCB) keyword (19) defines the physical organization of the data. It has four subkeywords:
  1. The LRECL= subkeyword (20) defines the logical record length.
  2. The RECFM= subkeyword (21) defines the record format. It may be fixed unblocked (F), fixed blocked (FB), variable unblocked (V), variable blocked (VB), variable blocked spanned (VBS), or undefined (U).
  3. The BLKSIZE= subkeyword (22) defines the physical block size.
  4. The DSORG= subkeyword indicates the data set organization. It may be sequential (PS), partitioned (PO), or direct (RR).
- The JOBLIB DD (23) is used to define the load module library where all programs of the job reside.
- You may use the concatenation of libraries to extend the search (24).
- As VSAM files are defined and cataloged by means of IDCAMS utilities, you only have to code the DSN= and DISP= keywords to address a VSAM data set (25).
- The only two mandatory keywords of the DD statement are DSN= and DISP=(26). The values for the other parameters are taken from the catalog. If the data set is not cataloged, you get a JCL error message.
- To access a particular member of a partitioned data set as a sequential file, you have to put the member name between parentheses following the data set name (27).
- For work data sets that last only for the duration of the job, you can use the temporary data set identifier &&name(28).
- To send the printed output to a system printer, use the SYSOUT= keyword (29), where you specify the printer output class name, or an asterisk, to default to the MSGCLASS= class stated in the JOB statement (30).
- You can include in-stream data in your job by coding a ′DD *′ statement (31). The data follow until a ′/*′ in column 1-2 is detected (32). The logical record length of these data is 80.
- The end of job is marked with a statement that only has ′//′ in columns 1-2(33).
- The NOTIFY=(34) parameter indicates which user will receive an appropriate message upon job termination, stating the completion code.

F.2 JCL Migration Considerations

As an explanation for those AIX users who have never had the opportunity to study JCL before now, we extract the information we need from the JCL to build an equivalent shell script.

The JCL in Figure 135 on page 138 is for a two-step job and is made up of:

- JOB card

  The JOB card is used by the operating system to recognize the user submitting the job and for accounting purposes. We need user identification to run our AIX application and can extract and use the user of the NOTIFY parameter on the JOB card for this purpose.

- JOBLIB card

  The JOBLIB card is used by the operating system to locate the executable programs for the two EXEC steps, PROCSTP1 and PROCSTP2. If all our application programs reside in an AIX directory with the same name as the application load library on the JOBLIB card, then the name of the library can be extracted. It is more common practice in AIX to use the PATH environment variable to locate specific applications.

- EXEC card

  We can extract two pieces of information from our EXEC cards: the step name and the program to execute.

- DD cards

  DD cards are used by the operating system to link the actual data sets with files being referenced by the executing program. Two types of DD cards are found: system DD cards and application DD cards. Most of the system DD cards can be ignored as they reference MVS system programs. The application DD cards must be converted as they refer 5o the files in our application.

  COBOL allows an AIX file name to be assigned to an environment variable. This environment variable can be used as an external file assignment within the COBOL program. All of these environment variables must be set prior to executing the program. See H.19, "Examples" on page 155 for an example of COBOL code using this method.

Our AIX shell script to perform the same function would be similar to that depicted in Figure 136 on page 142.

```
#!/bin/ksh
NOTIFY=user1
PROG1=/applic/inv/firstpgm
PROG2=/applic/inv/secndpgm
RUNCTL=/data/inv/ctlmembr
MYFILE=somesfsfile
MYOUTPUT=/data/tmp/tempfile
RUNRESLT=/data/tmp/runoutput
SYSPRINT=/data/tmp/prntfile
RUNTIME=/usr/local/bin.rtssfs
KEYTABF=/u/user1/keytab/jobkey
SECSET=/usr/local/bin/secSet
MYINPUT=/data/tmp/tempfile
MSTFILE=mstfile
NEWFILE=/data/inv/newfile
ENCINA_EXTFH_SFS=/.:/cics/sfs/yellow/
ENCINA_EXTFH_VOL=sfs_SFS_SERV

export NOTIFY PROG1 PROG2 RUNCTL MYFILE MYOUTPUT SYSPRINT RUNTIME
export KEYTABF SECSET ENCINA_EXTFH_SFS ENCINA_EXTFH_VOL
export MYINPUT MSTFILE NEWFILE RUNRESLT
#
# Run step 1 - program firstpgm
#
$SECSET $NOTIFY $KEYTABF $RUNTIME $PROG1 >>$RUNRESLT 2>>$RUNRESLT
#
# Run step 2 - program secndpgm depending on the outcome of step 1
#
if [ $? = 0 ]
then
    print "MYPROC - Step 1 completed date" >>$RUNRESLT
    $SECSET $NOTIFY $KEYTABF $RUNTIME $PROG2 >>$RUNRESLT 2>>$RUNRESLT
    print "MYPROC - Step 2 completed date RC = $?" >>$RUNRESLT
else
    echo "MYPROC has not completed succesfully"
    echo "check $RUNRESLT for ERRORS date"
    print "MYPROC - Step 1 aborted date RC = $?" >>$RUNRESLT
fi
```

Figure 136. MYPROC Shell Script

Some points to note about the shell script:

- SECSET - SFS files are under DCE security control, and application programs that need to access SFS files must be authenticated by the DCE security server. We used the secSet utility to do this. This tool can be found in SupportPac CA60. See A.5, "Hursley SupportPacs" on page 99 for additional information on SupportPacs.
- MSTFILE and MYFILE are both SFS files with SFS names *mstfile* and *somesfsfile*, respectively.

Appendix G. EBCDIC to ASCII Conversion

Normally the conversion of text from EBCDIC to ASCII is done automatically by the file transfer program. If your data contains binary or decimal fields as well as text fields, you will have to develop your own program to do the conversion. You can do the conversion on the MVS or AIX side. You just have to transfer your file in the binary mode. In this way the conversion table will be correct for your code page.

This appendix presents a simple conversion program to convert certain fields in a file from EBCDIC to ASCII. You can use this program as a sample and tailor it to your needs.

In the migration of our application we did not need to do any manual conversion from EBCDIC to ASCII.

G.1 Creating the Conversion Table File

To use the sample conversion program you must first create a file containing the conversion table. This file contains all possible values for 1 byte in 256 consecutive and ordered bytes. The first byte contains the ASCII value that corresponds to the EBCDIC hexadecimal value x′00′, and the last byte contains the ASCII value that corresponds to the EBCDIC hexadecimal value x′FF′.

The easiest way to create such a file is to generate a file with the EBCDIC hexadecimal values x′00′ to x′FF′ on your MVS system and then transfer it to your AIX system with the mode text, so that it gets converted. Figure 137 shows you a simple REXX program to generate the EBCDIC file.

```
/* REXX */
/* GENERATE OUTSTRING WITH BYTE 1 CONTAINING X′00′, BYTE 2 X′01′ UP */
/* TO BYTE 256 CONTAINING X′FF′                                     */
OUTSTRING = ′′
DO I = 0 TO 255
   OUTSTRING = OUTSTRING || D2C(I)
END

/* WRITE OUTSTRING TO A NEWLY ALLOCATED FILE CALLED USERID.OUT */
QUEUE OUTSTRING
QUEUE ′′
″DEL OUT″
″ALLOC FI(OUT) DA(OUT) RECFM(F B) LRECL(256)″,
″TRACK SPACE(1) UNIT(SYSDA) ″,
″NEW CATALOG BLKSIZE(80)″
″EXECIO * DISKW OUT (FINIS″
″FREE FI(OUT)″
```

Figure 137. REXX Program to Generate EBCDIC File

You can run this REXX program under any TSO authorized userid. After running it, you must transfer the generated *userid.out* to your AIX system in the mode text.

G.2 Conversion Program E2A

Once you have your conversion table file available on your AIX system, you can run your conversion program. Figure 138 shows you the sample source of the conversion program we used in our project.

```
/*------------------------------------------------------------------------*/
/*  -----------------                                                     */
/* | W A R N I N G : |  This program is provided on an ASIS basis.        */
/*  -----------------                                                     */
/*                                                                        */
/* This is a sample program to show you how to convert a file from EBCDIC to */
/* ASCII in which only certain fields are text.                           */
/* Author: Resident International Technical Support Center San Jose       */
/* Date  : 22nd July 1994                                                 */
/*                                                                        */
/* Input : This program receives the input data on stdin. Use the Unix    */
/*        cat command to process an existing file.                        */
/*                                                                        */
/* Output: This program writes it′s output to stdout. Use Unix redirection */
/*        (>file) to create a permanent file.                             */
/*                                                                        */
/* Arguments: The first argument must be the record length of the MVS     */
/*           file.                                                        */
/*           Argument 2 to n contain the starting position of each field  */
/*          you want converted from EBCDIC to ASCII and it′s length       */
/*           separated by a comma.                                        */
/*           Example: cat infile | e2a 80 0,10 15,5 39,40 >outfile        */
/*           This will convert three fields of the file infile with a     */
/*           record length of 80 (on the MVS system) and write the output */
/*           to the file outfile.                                         */
/*           Remember that to offset of the first character in a record is */
/*           always 0.                                                    */
/*                                                                        */
/*------------------------------------------------------------------------*/

/* specify all the necessary includes                                     */
#include <stdio.h>

/* The arguments are used for specifying the record length and the fields */
/* to be converted.                                                       */
main(int argc,char *argv[]) {
    char  *e2afile;           /* pointer to the file name of the file */
                                     /* containing the conversion table      */
    char  asciitab[255];      /* the in core conversion table */
    FILE  *fp;                /* pointer to the file containing the   */
                                     /* the conversion table                 */
    char  *in,**pEP,*pEnd;    /* pointers to character strings        */
    int   i,lrecl;            /* counter variable and record length   */
    char  start[10],length[10],end[2]=″,″;
    struct     column_struct {      /* structure for each text field       */
          int   start;
          int   length;
          struct column_struct   *next;
    };
    struct     column_struct    *first=NULL,*current=NULL,*previous=NULL ;
```

Figure 138 (Part 1 of 2). e2a.c Sample Program for EBCDIC to ASCII Conversion

```
    /* retrieve, open and read the file containing the conversion table */
    e2afile=getenv("E2AFILE");
    fp=fopen(e2afile,"r");
    if (fp==NULL) {
          printf("Invalid conversion file specified\n");
          printf("Check environment variable E2AFILE\n");
          exit(1);
    }
    fread(asciitab, sizeof(char), (size_t) 256, fp);
    fclose(fp);

    /* the argument 1 has to be the record length of MVS file */
    sscanf(argv[1],"%i",&lrecl);
    in=(char) malloc(lrecl);

    /* set up the table of fields to be converted */
    for(i=2;i<argc;i++) {
          /* allocate a pointer chained list for each field definition */
          previous=current;
          current=(struct column_struct *) malloc(sizeof(struct
          column_struct));
          if (first==NULL)
                first=current;
          else
                previous->next=current;

          /* parse the field definition, start and length are separated */
          /* by a comma                                                   */
          pEnd=end;
          pEP=&pEnd;
          current->start=strtol(argv[i],
          pEP,10);
          pEnd=*pEP;
          pEnd++;
          current->length=strtol(pEnd,NULL,10);
          current->next=NULL;    /* show that this is the last element */
                                       /* in our list                        */
    }

    /* process all the input records */
    while (fread(in,sizeof(char), (size_t) lrecl, stdin) == (size_t) lrecl)
    {
          /* loop through all fields to be converted */
          current=first;
          while (current) {
                /* convert all characters in a field */
                for(i=current->start;i<(current->start +
                current->length);i++) {
                      in[i]=asciitab[in[i]];
                }
                current=current->next; /* point to next field def. */
          }

          /* write record to stdout */
          fwrite(in, sizeof(char), (size_t) lrecl, stdout);
    }
}
```

Figure 138 (Part 2 of 2). e2a.c Sample Program for EBCDIC to ASCII Conversion

You can compile this program using the command shown in Figure 139 on page 146.

```
cc -o e2a e2a.c
```

Figure 139. Compile Command for e2a.c

Before you can run the e2a program you must make the location and name of your conversion table file generated by the REXX program known to it. There is no default location or name for this file. The environment variable *$E2AFILE* is read by e2a. If this variable contains an invalid file name or is not set, an error message is issued.

On each invocation of e2a you must specify the record length and at least the start and length of one text field in the file to be converted. You may specify as many fields as you like separated by blanks. The start and the length of each field must be separated by a comma. The correctness of these fields—that is, that they are valid numbers, are within the range of the record, do not specify the same field multiple times, and do not specify overlapping fields—is not checked. You may want to add this logic during your tailoring. You retrieve the record length from the definitions of your MVS data set. When specifying the field positions, remember that the first character in each record is in position zero.

You can run multiple conversions from an AIX shell script. Figure 140 shows you how to do this.

```
#!/bin/ksh
E2AFILE=/aix/directory/your.conversion.table
export E2AFILE
cat your input.file1 | e2a 80 0,10 19,1 >output.file1
cat your input.file2 | e2a 3245 0,1000 2009,156 2300,10 >output.file2
cat your input.file3 | e2a 10 0,10 >output.file3
```

Figure 140. Script for Running Multiple EBCDIC to ASCII Conversions

Run this script from any AIX userid. You can run this script in the background using the *&* option. In this way you can easily do the conversion of large files.

Appendix H. Interfacing COBOL with Encina SFS Files

This appendix is included for those users who may want to access Encina Structured File Server (SFS) files from COBOL batch programs, not using the CICS API. This process is continuously evolving and the information contained herewith should be viewed in that light. Future releases of products mentioned in this document may well have different function from those mentioned here.

H.1 Cobol and Encina SFS Overview

The sample programs were tested using Encina SFS V1.1.2.

You will access SFS files from COBOL by using the COBOL External File Handler. The External File Handler consists of some exits provided in the COBOL implementation. When invoked (for example, by an OPEN, CLOSE, READ, or WRITE), these exits call some other vendor′s code and access that other vendor′s file structures. In this case, Transarc has provided routines that are linked with a COBOL run time and, when invoked by an OPEN, CLOSE, READ, or WRITE inside your COBOL program, access SFS files instead of some other file structure.

Note

The discussion below applies when using the COBOL External File Handler (EXTFH) in conjunction with the Encina-provided routines to access the SFS. We refer to the function provided by the EXTFH but realize that the EXTFH just calls routines when it encounters an OPEN, WRITE, CLOSE, or READ and the routines that it calls to access the SFS are provided by Encina.

H.2 Documentation

The primary documentation on the COBOL External File Handler is supplied with the COBOL Compiler documentation, but that documentation does not discuss how the External File Handler relates to the routines provided with Encina to access SFS. Some additional documentation (in the form of a PostScript file) is supplied with Encina.

There are differences between SFS and VSAM. Read the appropriate sections of the online or hardcopy CICS/6000 documentation for a list of the differences. Some of the differences are also mentioned here if they provide clarity to this EXTFH discussion.

The *Encina ISAM Implementation and Extensions Guide* (SC23-2469-00), page 6-3, contains the descriptions of the status codes that are returned from COBOL verbs that work with the SFS (status codes that begin with 9).

H.3 Preparing the Run Time

1. Install COBOL
2. Build a new COBOL run time

   Enter the commands shown in Figure 141.

```
# export COBDIR=/usr/lib/cobol
# /usr/lpp/encina/scripts/build_rts32 -o rtssfs -d .
```

Figure 141. Building a New COBOL Run Time for SFS

Note

The COBDIR environment variable points to the directory where COBOL was installed. The **-o** is the name of the run time being produced, and **-d** indicates the directory that is to receive the run-time module. If you want to indicate the current directory, use the character **.**, or you may fully qualify the directory to receive the run-time module prepared for use with SFS.

At this point you have a COBOL run time (named rtssfs) that includes the Transarc-provided routines to access SFS.

H.4 Compiling the Program

Use the command in Figure 142 to compile your COBOL programs.

```
$ cob -uv testsfs.cbl
```

Figure 142. Compiling COBOL Program for Using SFS

Note

testsfs.cbl is the program name. The v is for verbose, the u indicates that an unlinked version (this is a .gnt module) should be produced for use with the run time.

H.5 Running the Program

Use the command shown in Figure 143 to execute your COBOL programs.

```
$ dce_login <principal> <password>
$ export ENCINA_EXTFH_VOL=sfs_SFS_SERV
$ export ENCINA_EXTFH_SFS=/.:/cics/sfs/<machine name>/
$ ./rtssfs testsfs    (this causes the runtime to execute the program)
```

Figure 143. Executing the COBOL SFS Program

Note

Make sure you have the ending "/" on the ENCINA_EXTFH_SFS environment variable. The testsfs is the COBOL .gnt module that was prepared above.

H.6 File Input and Output

COBOL allows four kinds of I/O:

1. Line sequential
2. Record sequential
3. Relative I/O
4. Indexed I/O.

The type of file organization is indicated in the ORGANIZATION IS clause of the SELECT statement in the FILE-CONTROL paragraph of the INPUT-OUTPUT SECTION. For example:

```
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT NOT OPTIONAL PART-DATA
               ASSIGN TO 'FILENAME'
               ORGANIZATION IS INDEXED
               ACCESS MODE IS RANDOM
               RECORD KEY IS PART-KEY
               STATUS IS RECORD-STATUS.
```

SELECT NOT OPTIONAL indicates that the file must exist before the program can be run.

Line sequential I/O (ORGANIZATION IS LINE SEQUENTIAL) provides a means of accessing regular sequential AIX files that have records delimited with a new line character (X'10') (as vi does automatically). Note that writing packed data to a LINE SEQUENTIAL file is not recommended.

Record sequential I/O (ORGANIZATION IS RECORD SEQUENTIAL) allows access to the SFS equivalent of the ESDS VSAM file.

Relative I/O (ORGANIZATION IS RELATIVE) allows access to the SFS equivalent of the VSAM RRDS file.

Indexed I/O (ORGANIZATION IS INDEXED) allows access to the SFS clustered file, which is the equivalent of the VSAM KSDS file.

H.7 Variable Length Files

Variable length records in SFS are implemented as one or more fixed length fields plus one or more variable length fields. Variable length records for use with CICS/6000 and batch COBOL usually contain one or more fixed length fields and end with one variable length fields. The ending variable length field is defined as varLenByteArray to SFS. CICS/6000 Schema definition SMIT panels support the definition of variable length file.

The field length header of the varLenByteArray field is made transparent to your CICS/6000 transaction and batch COBOL programs. Your program will see a variable length record, just like your program would see on the mainframe. C batch programs have great flexibility as to which parts of the record are returned and can see the entire record:

```
                 <<<<<<  RECORD LAYOUT  >>>>>

       logical user record:

         |-- 47 bytes  ------|-- variable record part -- .... ---|
         |------------  seen by batch COBOL ------------ .... ---|
         |------------ seen by CICS programs ----------- .... ---|

       physically defined to SFS:

                               +-- 4 byte header for varLenByteArray
                               |    field
                               v
         |--- 47 bytes  -----|---|--variable record part ---- .... ----|

         |--- batch Cobol ---|   |--- seen by batch Cobol --- .... ----|
         |-- by CICS progs --|   |--- seen by CICS programs - .... ----|
         |-- can be seen by batch C program ----------------- .... ----|
```

In order for SFS to write variable length records through a COBOL batch program you need to have either RECORDING MODE IS VARIABLE or RECORD VARYING IN SIZE FROM nn TO nnn in your FD along with a variable length record definition. Below is a sample COBOL FD definition for an SFS variable length record file:

```
        FD  OUTPUT-DATA
            RECORDING MODE IS VARIABLE
            DATA RECORD IS USER-REC.

        01  USER-REC.
            05 USER-KEY     PIC X(05).
            05 FILLER       PIC X(40).
            05 COUNTER      PIC 99.
            05 VAR-TBLE OCCURS 0 TO 50 TIMES
               DEPENDING ON COUNTER.
               10 VAR-DATA     PIC X(10).
```

Notes

1. The above example of the RECORDING MODE IS VARIABLE statement can be replaced by RECORD VARYING IN SIZE FROM 47 TO 547.
2. You must have a fixed part to your variable length record. You may **NOT** say RECORD IS VARYING IN SIZE FROM 0 to nnn.

H.8 Defining and Creating SFS Files

You can use the CICS/6000 SMIT panels to define files that will be used with the EXTFH.

You can use the `sfsadmin create...... file` command.

You may have the EXTFH create the SFS file for you by opening the file as output. Only the minimum number of fields will be created. The field names will be generated, that is, they will not be what you specified in the COBOL FD). The name of the index will be generated also, so be sure to put this name in your CICS/6000 FD if you create your files this way.

You may also use the sdt utility from SupportPac CA60 to create and maintain SFS files. See A.5, "Hursley SupportPacs" on page 99 for more information on how to obtain a SupportPac.

H.9 Record Limit

SFS files must have a `recordlimit` of `2147483647` or below, or the EXTFH will not be able to access your SFS data. The `recordlimit` of an SFS file may be changed after the file is created. To change the `recordlimit`, use the `sfsadmin set recordlimit` command.

H.10 Alternate Indexes

Alternate indexes built by the CICS/6000 Schema definitions cannot be accessed by EXTFH programs. EXTFH programs that work with a primary index will no longer work if the CICS/6000 Schema definitions are used to build a secondary index on that file. The problem is the form of the `sfsadmin` command that is used to build the secondary index. To be accessed by the EXTFH, the secondary index must be built with an alternate record layout as illustrated in Figure 144. Since CICS/6000 Schema panels do not define an alternate record layout but simply refer to a different field in the initial definition of the file, CICS/6000 transactions can access the secondary indexes created with the CICS/6000 Schema definitions, but an EXTFH program cannot.

In Figure 144, the file name is `payfile`, the secondary index name is `PaySecondIndex`, and the secondary index key field has a name of `AltKey`.

```
#!/bin/ksh
set -x
sfsadmin add index payfile PaySecondIndex \
              -alternaterecord 3 \
              Part1 byteArray 47 \
              AltKey byteArray 47 \
              TheRest byteArray 56 \
              1 AltKey
```

Figure 144. Building an Alternate Index

Note

If you have a file that already has an alternate index, and the file already contains data, you do not have to reload your data. You can just delete the index and add a new index as described above.

H.11 SFS Record Locking and Memory Growth

The default record locking with the EXTFH is to place read locks on records, even if the file is opened as INPUT. The accumulation of these locks in SFS′s memory causes the SFS to acquire memory as a program reads records. For a program reading several thousand records, this memory growth may consume all available memory, killing processes and ultimately killing SFS itself.

To avoid this problem for a program opening a file as INPUT and reading several thousand records, you can add `LOCK MODE IS MANUAL` to the `SELECT` clause and add `WITH NO LOCK` to the `READ` statement.

The EXTFH allows great flexibility with record and file locking. You may want to do some research in this area.

H.12 Reading and Writing Regular AIX Files

The EXTFH allows us access to SFS files. Here we take a brief look at access to regular AIX files.

H.12.1 Line Sequential Files

When using a COBOL run time prepared with the Encina-supplied scripts, the only access to regular AIX files is by using line sequential access. Line sequential has some limitations of which you should be aware. Records are delimited by X′0A′ (not good when the record contains binary or packed data). Packed field size appears to change based on the value of the field. Trailing spaces in the record are truncated.

H.12.2 Record Sequential Files

You can access record sequential files by building a run time with support for SFS ESDS files.

To remove the SFS ESDS support so that you can access record sequential AIX files:

1. Make a copy of `/usr/lpp/encina/scripts/build_rts32`.
2. Remove the `′-m sqfile=cobol_Extfh -m sqfilev=cobol_Extfh′` from the `cob` command at the end of that script.
3. Run your new script and generate an EXTFH run time that contains support for SFS KSDS, SFS RRDS, AIX line sequential, and AIX record sequential files.

H.13 Authentication and Permission

The authentication and permissions of the COBOL batch program will be inherited from the user running the program.

If you allow the EXTFH to create the file, the user will have to be authenticated as a principal with create authority for the SFS.

Running the program as `cell_admin` will have all necessary permissions, but you may want to set up permissions to run as a less powerful DCE principal.

If a file is created by the EXTFH and you later want to access the file with CICS/6000, you will have to `acl_edit` the file to allow CICS/6000 to access the file. Figure 145 shows a sample script file to set the file permissions.

```
#!/usr/bin/ksh
set -x
acl_edit /.:/cics/sfs/<hostname>/XXXXXX << EOF
modify unauthenticated:ADEIQRU
modify any_other:ADEIQRU
commit
exit
EOF
```

Figure 145. Setting SFS File Permissions

H.14 Transactional Access

SFS files may be accessed from batch while CICS/6000 is accessing the file. The COBOL EXTFH will NOT open a file transactionally, but, if desired, records can be locked from your batch program.

H.15 Debugging

The *Encina ISAM Implementation and Extensions Guide* (SC23-2469-00), page 6-3, contains descriptions of the status codes that are returned from COBOL verbs that work with SFS (status codes that begin with 9).

H.16 Environment Variables

The following environment variables must be set when using EXTFH and SFS:

- `COBDIR=` points to COBOL
- `ENCINA_EXTFH_VOL=` points the SFS-controlled volume that contains the file(s)

  The `ENCINA_EXTFH_VOL=` is used only when you allow the EXTFH to create files, but it is required to be present even if you are not going to create any files.
- `ENCINA_EXTFH_SFS=` name of the SFS (with a trailing "/").

  This variable can be omitted if you fully qualify the name of the SFS file in your program.

H.17 File References

You will probably have to change your COBOL program as to how it currently references SFS files.

If you want to reference SFS files from more than one SFS server at the same time, you will have to fully qualify the name of your SFS files ('/.:/cics/sfs/machine/file1')

You can refer to an SFS file from your COBOL program as follows:

```
SELECT file-name-1 ASSIGN TO literal-1
  (Where literal-1 is the SFS file name in quotes)

SELECT file-name-1 ASSIGN TO data-name-1
  (Where data-name-1 is a variable in your WORKING-STORAGE
  that contains the name of the SFS file)

SELECT file-name-1 ASSIGN TO EXTERNAL inputfile
  (Where inputfile is an environment variable)
```

When using the 'ASSIGN TO EXTERNAL inputfile' method, note the following rules:

- The environment variable you use may be uppercase or lowercase, but it may not exist as a variable in your program.
- Before executing your program, you must export the environment variable: `export dd_INPUTFILE=FixLengthFile`
- The environment variable must be in uppercase. FixLengthFile is the name of the actual SFS file.

For additional information on the `SELECT` statement see your COBOL language reference manual.

H.18 Executing a COBOL Program without a Run Time

Use the command shown in Figure 146 to prepare a COBOL program for execution without a run time (just specify the program name to execute the program). **Note** that this creates a huge executable object.

```
cob -x yourpgm.cbl -L /usr/lpp/encina/lib -L /usr/lib
-lEncSfsExtfh -lEncSfs -lEncina -ldce -lc_r -lpthreads
-m ixfile=cobol_Extfh -m ixfilev=cobol_Extfh
-m rlfile=cobol_Extfh -m rlfilev=cobol_Extfh
-m sqfile=cobol_Extfh -m sqfilev=cobol_Extfh
```

Figure 146. Creating an Executable COBOL Program (No Run Time)

H.18.1 Compiler Options

Note the following compiler options:

- Binary field alignment

  The IBMCOMP forces binary fields to be 2 or 4 bytes wide as they would be on the mainframe.

- Perform type

  Some COBOL programs may rely on certain PERFORM statement characteristics that are not defaults. If your perform statements do not respond as expected, you may want to add the following to the beginning of your source (starting in column 7): $SET PERFORM-TYPE"OSVS".

- CSI error

  If you are running with XREF as a compiler option and you receive a CSI error, try running with NOXREF. NOXREF is the default.

H.19 Examples

In this section we present examples of two COBOL program accessing SFS files and AIX sequential files.

H.19.1 Accessing Fixed Record Length Files

The sample program in Figure 147 on page 156 reads an AIX flat file and writes to an SFS KSDS file. We used an external file reference technique to reference the AIX flat file. To run this program, you need to have an environment variable set for the AIX flat file, **INPUTFILE=/dir1/data1**

Notice that even though we have lowercase inputfile specified in our SELECT statement, the COBOL compiler converts the file name to uppercase internally. Therefore, when you create the AIX environment variable, its name must be in uppercase.

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.     EXTREFDD.
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT NOT OPTIONAL PART-DATA ASSIGN  'FixLengthFile'
               ORGANIZATION IS INDEXED
               ACCESS MODE IS RANDOM
               RECORD KEY IS PART-KEY
               LOCK MODE IS MANUAL
               STATUS IS RECORD-STATUS.
           SELECT USER-DATA
               ASSIGN to external inputfile
               ORGANIZATION IS LINE SEQUENTIAL
               ACCESS MODE IS SEQUENTIAL
               STATUS IS RECORD-STATUS-2.

       DATA DIVISION.
       FILE SECTION.
       FD  USER-DATA
           DATA RECORD IS USER-REC.
       01  USER-REC.
           02 USER-KEY     PIC X(05).
           02 FILLER       PIC X(40).

       FD  PART-DATA
           DATA RECORD IS PART-REC.
       01  PART-REC.
           02 PART-KEY     PIC X(05).
           02 FILLER       PIC X(40).
       WORKING-STORAGE SECTION.
       01  WS-WORK-AREA.
           05  WS-COUNT                   PIC S9(8) COMP.
           05  EOF-INDICATOR              PIC X(4) VALUE 'FALS'.
      *
       01  RECORD-STATUS              PIC XX VALUE '00'.
       01  RECORD-STATUS-2            PIC XX VALUE '00'.
```

Figure 147 (Part 1 of 2). Sample Sequential SFS Program

```
       PROCEDURE DIVISION.
           PERFORM OPEN-THE-FILES.
      *
           MOVE ZERO TO WS-COUNT.
           PERFORM PROCESS-RECORD THRU
                 PROCESS-RECORD-EXIT
               UNTIL EOF-INDICATOR = 'TRUE'.
           DISPLAY 'end of sequential file, number of records = '
               WS-COUNT.
           CLOSE PART-DATA.
           CLOSE USER-DATA.
           GOBACK.


       OPEN-THE-FILES.
           OPEN I-O PART-DATA.
           IF RECORD-STATUS NOT EQUAL '00'
               DISPLAY 'sfs file open failed status = '
                        RECORD-STATUS
               GOBACK.
           OPEN INPUT USER-DATA.
           IF RECORD-STATUS-2 NOT EQUAL '00'
               DISPLAY 'sequential file open failed status = '
                        RECORD-STATUS-2
               GOBACK.

       PROCESS-RECORD.
           READ USER-DATA
               AT END
                 MOVE 'TRUE' TO EOF-INDICATOR
                 GO TO PROCESS-RECORD-EXIT.
           IF RECORD-STATUS-2 NOT EQUAL '00'
               DISPLAY 'read of sequential file failed, status = '
                        RECORD-STATUS-2
                        ' KEY = ' WS-COUNT
               GOBACK.

           MOVE USER-REC TO PART-REC.

           WRITE PART-REC.
           ADD +1                      TO WS-COUNT.
           IF WS-COUNT GREATER THAN 150
               DISPLAY '150 records written, ending'
               GOBACK.
           IF RECORD-STATUS EQUAL '22'
               DISPLAY 'Attempt to add duplicate record, record # = '
                         WS-COUNT
               GO TO PROCESS-RECORD-EXIT.
           IF RECORD-STATUS NOT EQUAL '00'
               DISPLAY 'write of sfs file failed, status = '
                        RECORD-STATUS
                        ' RECORD # = ' WS-COUNT
               GOBACK.
       PROCESS-RECORD-EXIT.
           EXIT.
```

Figure 147 (Part 2 of 2). Sample Sequential SFS Program

H.19.2 Accessing Variable Record Length Files

The sample program in Figure 148 on page 158 inputs an SFS KSDS variable record length file and outputs two SFS KSDS variable length files.

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.     VARWRITE.
       ENVIRONMENT DIVISION.
       INPUT-OUTPUT SECTION.
       FILE-CONTROL.
           SELECT NOT OPTIONAL PART-DATA
               ASSIGN TO 'sfs_var_outfile6'
               ORGANIZATION IS INDEXED
               ACCESS MODE IS RANDOM
               RECORD KEY IS PART-KEY
               LOCK MODE IS MANUAL
               STATUS IS RECORD-STATUS.

           SELECT OPTIONAL PART-DATA-7
               ASSIGN TO 'sfs_var_outfile7'
               ORGANIZATION IS INDEXED
               ACCESS MODE IS RANDOM
               RECORD KEY IS PART-KEY-7
               LOCK MODE IS EXCLUSIVE
               STATUS IS RECORD-STATUS-7.

           SELECT NOT OPTIONAL USER-DATA ASSIGN 'sfs_var_input'
               ORGANIZATION IS INDEXED
               ACCESS MODE IS SEQUENTIAL
               RECORD KEY IS USER-KEY
               LOCK MODE IS MANUAL
               STATUS IS RECORD-STATUS-2.

       DATA DIVISION.
       FILE SECTION.

       FD  USER-DATA DATA RECORD IS USER-REC.
       01  USER-REC.
           02 USER-KEY     PIC X(05).
           02 FILLER       PIC X(40).
           02 LEO-CR-O     PIC 99.
           02 VAR-TBLE-O OCCURS 0 TO 50 TIMES
              DEPENDING ON LEO-CR-O.
              05 VAR-DATA-O     PIC X(10).

       FD  PART-DATA-7
           RECORD VARYING IN SIZE FROM 47 TO 547
           DATA RECORD IS PART-REC-7.
       01  PART-REC-7.
           02 PART-KEY-7     PIC X(05).
           02 FILLER-7       PIC X(40).
           02 LEO-CR-7       PIC 99.
           02 VAR-TBLE-7 OCCURS 0 TO 10 TIMES
              DEPENDING ON LEO-CR-7.
              05 VAR-DATA-7     PIC X(100).

       FD  PART-DATA
           RECORDING MODE IS VARIABLE
           DATA RECORD IS PART-REC.
       01  PART-REC.
           02 PART-KEY     PIC X(05).
           02 FILLER       PIC X(40).
           02 LEO-CR       PIC 99.
           02 VAR-TBLE OCCURS 0 TO 50 TIMES
              DEPENDING ON LEO-CR.
              05 VAR-DATA     PIC X(10).
```

Figure 148 (Part 1 of 3). Sample SFS KSDS Program

```
        WORKING-STORAGE SECTION.
        01  WS-WORK-AREA.
            05  WS-COUNT                   PIC S9(8) COMP.
            05  EOF-INDICATOR              PIC X(4) VALUE 'FALS'.
            05  WS-FLAG                    PIC X VALUE 'Y'.
       *
        01  RECORD-STATUS-FIELDS.
            05 RECORD-STATUS.
               10 STATUS1                     PIC 9.
               10 STATUS2                     PIC 9.
               10 STATUS2R REDEFINES STATUS2  PIC 99 COMP-X.
            05 OUTFILE-STATUS              PIC 999.
       *
            05 RECORD-STATUS-7.
               10 STATUS1-7                     PIC 9.
               10 STATUS2-7                     PIC 9.
               10 STATUS2R-7 REDEFINES STATUS2-7  PIC 99 COMP-X.
            05 OUTFILE-STATUS-7              PIC 999.

        01  RECORD-STATUS-2             PIC XX VALUE '00'.

        PROCEDURE DIVISION.
            PERFORM OPEN-THE-FILES.
       *
            MOVE ZERO TO WS-COUNT.
            PERFORM PROCESS-RECORD THRU
                  PROCESS-RECORD-EXIT
                UNTIL EOF-INDICATOR = 'TRUE'.
            DISPLAY 'end of sequential file, number of records = '
                WS-COUNT.
            CLOSE PART-DATA.
            CLOSE USER-DATA.
            GOBACK.


        OPEN-THE-FILES.
            OPEN INPUT USER-DATA.
            IF RECORD-STATUS-2 NOT EQUAL '00'
                DISPLAY 'sequential file open failed status = '
                         RECORD-STATUS-2
                GOBACK.
            OPEN I-O PART-DATA.
            IF RECORD-STATUS NOT EQUAL 0
                DISPLAY 'isef6 open failed status = '
                         RECORD-STATUS
            IF STATUS1 EQUAL 9
                MOVE STATUS2R TO OUTFILE-STATUS
                DISPLAY 'isef6 open failed status = '
                         OUTFILE-STATUS
                GOBACK.

            OPEN OUTPUT PART-DATA-7.
            IF RECORD-STATUS-7 NOT EQUAL 0
                DISPLAY 'isef7 open failed status = '
                         RECORD-STATUS-7
            IF STATUS1-7 EQUAL 9
                MOVE STATUS2R-7 TO OUTFILE-STATUS-7
                DISPLAY 'isef7 open failed status = '
                         OUTFILE-STATUS-7
                GOBACK.
```

Figure 148 (Part 2 of 3). Sample SFS KSDS Program

```
       PROCESS-RECORD.
           READ USER-DATA
               AT END
                 MOVE 'TRUE' TO EOF-INDICATOR
                 GO TO PROCESS-RECORD-EXIT.
           IF RECORD-STATUS-2 NOT EQUAL '00'
               DISPLAY 'read of sequential file failed, status = '
                        RECORD-STATUS-2
                        ' KEY = ' WS-COUNT
               GOBACK.

           MOVE USER-REC TO PART-REC.
           MOVE USER-REC TO PART-REC-7.
           ADD +1                      TO WS-COUNT.

           WRITE PART-REC.
           WRITE PART-REC-7.

           MOVE 'N' TO WS-FLAG.
           IF WS-COUNT GREATER THAN 9
               DISPLAY '10 records written, ending'
               GOBACK.
           IF RECORD-STATUS EQUAL '22'
               DISPLAY 'Attempt to add duplicate record, record # = '
                         WS-COUNT
               GO TO PROCESS-RECORD-EXIT.
           IF RECORD-STATUS NOT EQUAL '00'
               DISPLAY 'write of sfs file failed, status = '
                        RECORD-STATUS
                        ' RECORD # = ' WS-COUNT
               GOBACK.
       PROCESS-RECORD-EXIT.
           EXIT.
```

Figure 148 (Part 3 of 3). Sample SFS KSDS Program

Glossary

A

aixterm. A terminal emulator provided by AIXwindows to run programs that do not know how to use windows.

API. Application programming interface. A set of calling conventions defining how a service is invoked through a software package.

application. (1) The use to which an information processing system is put: for example, a payroll application, an airline reservation application, a network application. (2) A collection of software components used to perform specific types of user-oriented work on a computer.

authentication. In computer security: (1) verification of the identity of a user or the user′s eligibility to access an object. (2) Verification that a message has not been altered or corrupted.

authorization. In computer security: (1) The right granted to a user to communicate with or make use of a computer system. (2) An access right. (3) The process of granting a user either complete or restricted access to an object, resource, or function.

B

background process. (1) A process that does not require operator intervention but can be run by a computer while the workstation is used to do other work. (2) In the AIX operating system, a mode of program execution in which the shell does not wait for program completion before prompting the user for another command.

batch. An accumulation of data to be processed with little or no user intervention.

breakpoint. A point in a computer program where execution may be halted. A breakpoint is usually at the beginning where halts, caused by external intervention, are convenient for resuming execution.

C

cache. A buffer storage that contains frequently accessed instructions and data; it is used to reduce access time.

CASE. Computer-aided software engineering. A set of tools or programs to help develop complex applications.

client/server computing. A form of distributed processing in which the task required to be processed is accomplished by a client portion that requests services and a server portion that fulfills those requests. The client and server remain transparent to each other in terms of location and platform.

closed shop. (1) Pertaining to the operation of a computer facility in which most productive problem programming is performed by a group of specialists rather than by the problem originators. The use of the computer itself may also be described as closed shop if full-time trained operators, rather than users or programmers, serve as the operators.

code page. An assignment of graphic characters and control functions to all code points; for example, assignment of characters and meanings to 256 code points for an 8-bit code, assignment of characters and meanings to 128 code points for a 7-bit code.

connection. In data communication, an association established between functional units for conveying information.

copybook. COBOL source code maintained in a separate file outside the main program. It is included into the main program at compile time.

D

database. (1) A collection of interrelated data stored together with controlled redundancy according to a scheme to serve one or more applications. (2) All of the data files stored in a system. (3) A set of data stored together and managed by a database management system.

DCE. Distributed Computing Environment. De facto standard adopted by the industry as a standard for distributed computing. DCE allows computers from a variety of vendors to communicate transparently and to share resources such as computing power, files, printers, and other objects in the network.

DDL. Data definition language. A language for describing data and its relationships in a database.

debug. To detect, to locate, and to eliminate errors in computer programs.

directory. A repository of information about objects, functions, or services and a set of services to manipulate the repository.

distributed program link. A CICS application program can pass control to a second program, which runs as a called subroutine, returning control on completion to the point of invocation in the first program, whereas the called program resides on a different CICS system.

downsizing. Migrating a fully functional application from a mainframe environment to a workstation.

E

Encina. Enterprise Computing In a New Age. A set of DCE-based products from Transarc Corporation that are available on the RISC System/6000. Encina is a family of online transaction processing products and includes:

- Encina Toolkit Executive
- Encina Server
- Encina Structured File Server (SFS)
- Encina Peer-to-Peer Communication Executive (PPC).

environment. The collective hardware and software configuration of a system.

environment variable. In the AIX operating system: (1) A variable that describes the operating environment of the process. (2) A variable that is included in the current software environment and is therefore available to any called program that requests it.

F

flag. (1) A variable indicating that a certain condition holds. (2) Any of various types of indicators used for identification; for example, a woodmark. (3) A character that signals the occurrence of some condition, such as the end of a word.

function shipping. Allows CICS application programs to access files owned by other CICS systems.

G

gateway. (1) A functional unit that interconnects two computer networks with different network architectures. A gateway connects networks or systems of different architectures. (2) In the AIX operating system, an entity that operates above the link layer and translates, when required, the interface and protocol used by one network into those used by another distinct network. (3) In TCP/IP, a device used to connect two systems that use the same or different communication protocols. (4) In the IBM Token-Ring Network, a device and its associated software that connect a local area network to another local area network or a host that uses different logical link protocols.

glassbox test. A form of testing in which the structural composition of the object to be tested is known to the tester (also known as whitebox test).

GUI. Graphical user interface. A style of user interface that replaces the character-based screen with an all-points-addressable, high-resolution graphics screen that uses windows to display multiple applications at the same time while allowing user input by means of a keyboard or a pointing device such as a mouse, pen, or trackball.

H

host. (1) In a computer network, a computer providing services such as computation, database access, and network control functions. (2) The primary or controlling computer in a multiple computer installation.

I

integrity. The protection of systems, data and programs from inadvertent or malicious destruction or alteration.

intersystem communication. (1) Transfer of data between systems by means of data exchange or data interchange. (2) In CICS/VS, a subset of SNA formats and protocols that governs the interactions of transaction processing systems. By defining how transaction programs communicate, even when they are under control of different transaction processing systems, intersystem communication allows construction of a distributed transaction processing application using individual transaction programs as building blocks.

J

JCL. Job control language. A control language used to identify a job to an operating system and to describe the job′s requirements.

job. A user-defined unit of work that is to be accomplished by a computer. Loosely, the term job is sometimes used to refer to a representation of a job. This representation may include a set of computer programs, files, and control statements.

K

Korn shell. An interactive command interpreter and command programming language.

L

LAN. Local area network. A computer network located on a user′s premises within a limited geographical area.

layer. In a network architecture, a group of services that is complete from a conceptual point of view, that is, one out of a set of hierarchically arranged groups, and that extends across all systems that conform to the network architecture.

M

macro. A possibly parameterized specification for a statement sequence to replace a macro call.

mainframe. A computer, usually in a computer center, with extensive capabilities and resources to which other computers may be connected so that they can share facilities.

migration. (1) The process of moving data from one computer system to another without converting the data. (2) Installation of a new version or release of a program to replace an earlier version or release.

N

network. A configuration of data processing devices and software connected for information interchange.

O

OLTP. Online transaction processing. A style of computing that supports interactive applications in which requests submitted by terminal users are processed as soon as they are received. Results are returned in a relatively short period of time. An online transaction processing system supervises the sharing of resources for processing multiple transactions at the same time, minimizes compute time and duration of locks, and separates user thinking-time from the use of storage and other resources.

P

package. Contains an internal representation of the SQL statements in a program for a DB2/6000 program and is stored in the database.

pipe. (1) To direct data so that the output from one process becomes the input to another process. The standard output of one command can be connected to the standard input of another with the pipe operator (|). (2) A one-way communication path between a sending process and a receiving process.

plan. Contains an internal representation of the SQL statements in a program for a DB2 MVS program and is stored in the database.

portability. The ability to move application software from one system for use on another system. Perfect portability would permit such movement without modification to those components.

principal. An entity that can be authenticated (its network identity verified) by the DCE Security Service. It is represented as an entry in the Security Service registry database, where information about principals (and other objects) is stored. Principals include users, servers, machines, and cells.

privilege. In SQL, a capability given to a user by the processing of a GRANT statement.

profile. (1) Data that describes the significant characteristics of a user, a group of users, or one or more computer resources. (2) In computer security, a description of the characteristics of an entity to which access is controlled.

project. An undertaking with prescribed objectives, magnitude, and duration.

protocol. (1) A formal set of conventions governing the format and control of data. (2) A set of procedures or rules for establishing and controlling transmissions from a source device or process to a target device or process.

PWS. Programmable workstation. A workstation that has some degree of processing capability and allows a user to change its functions.

Q

qualifier. (1) A modifier that makes a name unique.

queue. A list constructed and maintained so that the next data element to be retrieved is the one stored first.

R

referential integrity. The state of a database in which all values of foreign keys are valid.

region. An entity that owns its own set of CICS resources.

resource manager. A software program that maintains the state of resources and provides access and control to them through an API. A resource can be a device as well as a program or object.

S

script. A text file that contains instructions to be executed by an interpreter.

server. Any computing resource dedicated to responding to client requests. Servers can be linked to clients through LANs or WANs to perform services on behalf of multiple clients.

sequential data set. A data set whose records are organized on the basis of their successive physical positions.

session. In a network architecture, for the purpose of data communication between functional units, all of the activities that take place during the establishment, maintenance, and release of the connection.

shell. A software interface between a user and the operating system of a computer. Shell programs interpret commands and user interactions on devices such as keyboards, pointing devices, and touch-sensitive screens and communicate them to the operating system.

SQL. Structured query language. SQL started as IBM′s query language for DB2. SQL became so popular with users and vendors outside IBM that ANSI adopted a version of SQL as a U.S. standard in 1986. A year later ISC gave SQL formal international standard status.

stanza. A group of lines in a file that together have a common function or define a part of the system.

subsystem. A secondary or subordinate system, usually capable of operating independently of, or asynchronously with, a controlling system.

symbolic link. Allows access to data in other file systems from a new file name. The symbolic link is a regular file that contains a path name.

T

TCP/IP. Transmission Control Protocol/Internet Protocol. A set of communications protocols that support peer-to-peer connectivity for both local and wide area networks.

terminal. A functional unit in a system or communication network at which data may enter or leave.

transaction. A unit of processing (consisting of one or more application programs) initiated by a single request. A transaction may require the initiation of one or more tasks for its execution.

transaction manager. Provides a total environment for transactional applications. In addition to transaction manager functions, it provides services to aid development, execution, and operation of transactional applications.

transaction processing. A style of computing that supports interactive applications in which requests submitted by users are processed as soon as they are received. Results are returned to the requester in a relatively short period of time. A transaction processing system supervises the sharing of resources for processing multiple transactions at the same time.

transaction routing. Allows a user to enter a remote transaction identifier at a terminal, and the transaction runs as though it were owned by the local system.

two-phase commit. A database protocol that is used to ensure uniform transaction commit or abort in a distributed data environment between two or more participants. The protocol consists of two phases: the first to reach a common decision, and the second to implement the decision.

TX. Within the Distributed Transaction Processing Model adopted by X/Open, the interface between the application and the transaction manager.

U

UOW. Unit of work. In advanced program-to-program communication, the amount of processing that is started directly or indirectly by a program on the source system.

V

vi. A full screen editor common among UNIX platforms.

W

workstation. A configuration of input and output equipment at which an operator works. A terminal or microcomputer, usually one that is connected to a mainframe or to a network at which a user can perform applications.

X

XA. Within the Distributed Transaction Processing Model adopted by X/Open, the interface between the transaction manager and the resource managers.

List of Abbreviations

2PC	two-phase commit
ACL	access control list
AIC	AIX Interface Composer
AIX	Advanced Interactive Executive
AMA	application migration aid
AOR	application owning region
API	application programming interface
APPL	application
ASCII	American Standard Code for Information Interchange
ATI	automatic task initiation
BLL	base locator for linkage
BMS	basic mapping support
CAF	call attachment facility
CASE	computer-aided software engineering
CDRSC	cross-domain resources
CDS	Cell Directory Service (OSF/DCE)
CICS	Customer Information Control System
COBOL	Common Business Oriented Language
CPI-C	Common Programming Interface for Communications
CSD	CICS system definition
CUA	Common User Access
DB2	DATABASE 2
DBRM	database request module
DCB	data control block
DCE	Distributed Computing Environment
DCL	declare
DCL	data control language
DCT	destination control table
DD	data definition
DDCS	Distributed Database Connection Services
DDF	distributed data facility
DDL	data definition language
DLC	data link control
DLI	Data Language I
DM	distribution management
DME	Distributed Management Environment
DML	data manipulation language
DOS	Disk Operating System
DRDA	Distributed Relational Database Architecture
DSMIT	Distributed System Management Tool
DTP	distributed transaction processing
EBCDIC	Extended binary-coded decimal interchange code
ECI	external call interface
EDF	execution diagnostic facility
ESA	Enterprise Systems Architecture
FAQ	frequently asked question
FTP	file transfer program
FTP	file transfer protocol
GDDM	Graphical Data Display Manager
hcon	IBM AIX 3270 host connection program/6000
IBM	International Business Machines Corporation
ISC	intersystem communication
ISPF/PDF	Interactive System Productivity Facility/Program Development Facility
IT	information technology
ITSO	International Technical Support Organization
IVP	installation verification program
IXF	integrated exchange format
JES	job entry subsystem
JCL	job control language
LAN	local area network
LPP	licensed program product
LU	logical unit
MB	megabyte
MSR	magnetic slot reader

MVS	Multiple Virtual System
OLTP	online transaction processing
OPC	operations, planning, and control
OSF	Open Systems Foundation
OS/2	Operating System/2
PA	program access (key)
PDS	partitioned data set
PLI	Programming Language I
PPC	peer-to-peer communication
PU	physical unit
PWS	programmable workstation
QMF	Query Management Facility
RDBMS	relational database management system
RDO	Resource Definition Online
REXX	Restructured Extended Executor
RISC	Reduced Instruction Set Computer
RM	resource manager
RPC	remote procedure call
RPQ	request for price quotation (IBM custom built machines or features)
RSCS	remote spooling communications subsystem
RTS	run-time system
SAA	Systems Application Architecture
SCS	SNA character string
SDA	Screen Design Aid
SDF	Screen Definition Facility
SFS	Structured File Server
SIT	system initialization table
SMIT	System Management Interface Tool
SNA	Systems Network Architecture
SQL	structured query language
tar	tape archive
TCP/IP	Transmission Control Protocol/Internet Protocol
TCT	terminal control table (CICS)
TD	transaction definition
TD	transient data
TM	transaction manager
TOR	terminal owning region
TPN	transaction program name
TSO	Time Sharing Option
vi	visual screen-based editor
VM	Virtual Machine
VNET	Virtual Node Exchange Transmission
VSAM	Virtual Storage Access Method
VTAM	Virtual Telecommunications Access Method
WAN	wide area network
WCC	write control character
WD	workstation definition
WWW	World Wide Web

Index

ITSO Technical Bulletin Evaluation RED000

MVS to AIX Application Migration Cookbook

Publication No. GG24-4375-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

What other topics would you like to see in this Bulletin?

What other Technical Bulletins would you like to see published?

Comments/Suggestions: (THANK YOU FOR YOUR FEEDBACK!)

Name

Address

Company or Organization

Phone No.

ITSO Technical Bulletin Evaluation **RED000**
GG24-4375-00

Cut or Fold
Along Line

Fold and Tape **Please do not staple** Fold and Tape

NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES

BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department 985, Building 657
P.O. BOX 12195
RESEARCH TRIANGLE PARK NC
USA 27709-2195

Fold and Tape **Please do not staple** Fold and Tape

GG24-4375-00

Cut or Fold
Along Line

GG24-4375-00