IBM

International Systems Centers

Office Systems Interconnection: Guide to Connecting non-DIA Systems to DISOSS

GG24-1604

Office Systems Interconnection: Guide to Connecting non-DIA Systems to DISOSS

Document Number GG24-1604

March 1984

Rob van Olmen, IBM Netherlands
Laurence Barker, IBM UK
Warwick Wright, IBM New Zealand
David Harding, IBM Australia
Gordon Hay, Raleigh ISC

Project Adviser: Gordon Hay
Raleigh International Systems Centre
PO Box 12195, Dept 985/622
Research Triangle Park
NC 27709, USA

The information contained in this document has not been submitted to any formal IBM test and is distributed on an 'As Is' basis without any warranty either expressed or implied. 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 samples described in this material are presented for illustrative purposes only and are not intended to be implemented as described.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used instead.

It is possible that this material may contain reference to, or information about, IBM products (machines and programs), programming or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such IBM products, programming or services in your country.

First Edition (March 1984)

Publications are not stocked at the address below; requests for copies of IBM publications should be made to your IBM representative or to the IBM branch office serving your locality.

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

Raleigh International Systems Centre
IBM Corporation (985/B622-3)
PO Box 12195
Research Triangle Park, NC 27709, USA

IBM may use or distribute any of the information you supply without incurring any obligation to you. You may, of course, continue to use the information you supply.

ABSTRACT

DISOSS/370 provides document library and distribution services for a range of intelligent subsystems (DISOSS/8100, Displaywriter, 5520, Scanmaster and DISOSS/PS) which support the IBM document handling architectures DIA and DCA.

This book examines the feasibility of extending these document filing and distribution functions to the various other intelligent devices that do not support DIA and DCA; the assumption is that, although these devices are often not primarily office systems, their users would nevertheless benefit from being able to use a company-wide electronic mail distribution system.

The purpose is to show what can be done with currently available products, and the book includes an example showing how documents can be exchanged between DISOSS and PROFS.

PREFACE

This document is the result of a Residency Project conducted at the Raleigh International Systems Centre. The purpose of the project was to study the problem of connecting non-DIA systems to DISOSS, and to develop design guidelines for the use of anyone wishing to implement such a connection.

There are four major parts:

- "Introduction and Design Overview" on page 3 describes the objectives of the project, summarises the design options we considered, and provides an overview of the suggested system design.
- "System Structure" on page 19 describes in detail the components of the proposed system.
- "Sample System Implementation" on page 27, "System Definitions for the Sample Implementation" on page 47, and "Communicating Between PROFS and DISOSS" on page 53 describe our sample implementation of the proposed system.
- "Sample System Components" on page 67, "API and DIU-build Subroutines" on page 105 and "Sample EXECs for the CMS and PROFS User" on page 155 contain listings of the programs used in the sample implementation.

Anyone contemplating the possibility of attaching non-DIA systems to DISOSS should find the "Introduction and Design Overview" on page 3 useful. The remaining chapters are intended for systems designers and programmers responsible for the design and implementation of such interfaces.

RELATED PUBLICATIONS

SC30-3096	DISOSS/370 Version 3 Application Programming
GG24-1614	DISOSS Application Interface: Programming Guidelines

ACKNOWLEDGEMENTS

Most of the sample programs shown in this book depend on a set of generalised DISOSS API subroutines; these were designed and developed by Martin Hibbert of Technical Support, IBM UK, and are described in detail in **DISOSS Application Interface: Programming Guidelines, GG24-1614.**

ABBREVIATIONS

Certain terms widely used in this document should be understood as follows:

JES Refers to Job Entry Subsystems in general, and should be taken to include VSE/POWER, OS/VS1 RES, MVS/JES2 and MVS/JES3, except where otherwise stated.

Box X Refers to the subsystem wishing to communicate with DISOSS. Where there is no need to distinguish between the various real systems (VM/SP, S/34, DPPX etc.), we will often use the term 'Box X' to apply to them all.

L2DCA Refers to the Level 2 Document Content Architecture. This is also known as the Final Form Text DCA, or FFT, and is defined in **Document Content Architecture: Final-Form-Text Reference, SC23-0757.**

SYSTEM DESIGN 1

1.0 Introduction and Design Overview 3
1.1 Design Objectives 4
1.2 Logic of the Design Process 4
1.2.1 Distribution of Function 5
1.2.2 End-User Interface 5
1.2.3 Available Tools 5
1.2.4 DISOSS/PS for Interactive Functions 6
1.2.5 RJE for Batch Functions 8
1.2.6 Moving Documents from CICS to JES 9
1.2.7 Moving Documents from JES to CICS 9
1.2.7.1 Direct SNA Session 9
1.2.7.2 SNA Relay Program 10
1.2.7.3 JES External Writer 11
1.2.7.4 Shared Dataset 12
1.2.8 CICS Application to Access DISOSS API 12
1.3 Overview of the System Structure 16

2.0 System Structure 19
2.1 Interactive Communication 19
2.2 Document Transfer from DISOSS to Box X 19
2.3 Document Transfer from Box X to DISOSS 19
2.3.1 Major Components 21
2.3.1.1 Box X Job Submission 21
2.3.1.2 Batch Program 21
2.3.1.3 CICS Program DBTSON1 21
2.3.1.4 CICS Transaction DBTM 23
2.3.1.5 CICS Transaction DBTS 23
2.3.1.6 CICS Transaction DBTR 23

SAMPLE IMPLEMENTATION 25

3.0 Sample System Implementation 27
3.1 Moving Documents from DISOSS to Box X 28
3.2 Moving Documents from Box X to DISOSS 29
3.2.1 Use of the DISOSS API 29
3.2.1.1 General Remarks 29
3.2.1.2 Our Use of the API. 30
3.2.2 Document Transformations and Translations 34
3.2.2.1 Datastream Transformations 34
3.2.2.2 Character Translations 36
3.2.2.3 Overview of Transformations and Translations 39
3.2.3 Batch-CICS Interface 40
3.2.4 Components of the Box X to DISOSS Facility. 44
3.2.4.1 DBTBAT1 44
3.2.4.2 DBTMOV1 44
3.2.4.3 DBTOPN1 44
3.2.4.4 DBTCLS1 44
3.2.4.5 DBTMST1 45
3.2.4.6 DBTTRN1 45
3.2.4.7 DBTSND1 45
3.2.4.8 DBTRSP1 46
3.2.4.9 DBTSON1 46
3.2.4.10 DBTCLN1 46
3.2.4.11 DBTSOF1 46

4.0 System Definitions for the Sample Implementation 47
4.1 CICS Tables 47
4.1.1 File Control Table 47
4.1.2 Program Control Table 47
4.1.3 Program List Table 48
4.1.4 Program Processing Table 48
4.1.5 System Initialisation Table 49
4.2 VSAM Dataset Definitions 49
4.3 DISOSS Table Definitions (Box X to DISOSS) 50
4.3.1 Translate Tables 50
4.3.2 Host User Profile 50
4.4 DISOSS Table Definitions (DISOSS to Box X) 50
4.4.1 Host User Profile 50
4.4.2 Printer Description Table 51
4.5 Print/Format Procedure 51
4.6 Improvements and Alternative Options 51

5.0 Communicating Between PROFS and DISOSS 53
5.1 Overview 53
5.2 Accessing DISOSS/PS from PROFS 54
5.3 Sending a PROFS document to DISOSS 55
5.4 Sending a PROFS note to DISOSS 57
5.4.1 Forwarding an Existing Note 57
5.4.2 The DBTNOTE EXEC 59
5.4.3 Creating and Sending a New Note 59
5.5 Changing a DCF file to 1403 Format 60
5.6 Loading a DISOSS/PS Document to the A-disk and then to PROFS . 61

APPENDICES 65

A.0 Sample System Components 67
A.1 General Remarks 67
A.2 Software Used to Test the Design 67
A.3 Source Listings 68
A.3.1 DBTVSQ Common Block 69
A.3.2 DBTOC Common Block 71
A.3.3 DBTDOCIN Catalogued Procedure 72
A.3.4 DBTBAT1 Program Source 73
A.3.5 DBTMOV1 Program Source 77
A.3.6 DBTOPN1 Program Source 80
A.3.7 DBTCLS1 Program Source 81
A.3.8 DBTMST1 Program Source 82
A.3.9 DBTTRN1 Program Source 83
A.3.10 DBTSND1 Program Source 88
A.3.11 DBTRSP1 Program Source 92
A.3.12 DBTSON1 Program Source 94
A.3.13 DBTCLN1 Program Source 96
A.3.14 DBTSOF1 Program Source 97
A.3.15 DBTTRT01 Translate Table 99
A.4 Improvements and Alternatives 103
A.5 Simultaneous CICS/Batch Access to Shared Dataset 103

B.0 API and DIU-build Subroutines 105
B.1 Assembler Control Blocks 105
B.1.1 APICOM 105
B.1.2 APIDPR2 107
B.1.3 APIRET 109
B.1.4 APIREGS 109
B.2 PL/I Control Blocks 110
B.2.1 APICOMP 110

B.2.2 APIDPRP 111
B.2.3 APIDPRP2 112
B.2.4 APIRETP 113
B.2.5 APIRETP2 113
B.3 Assembler Macro 114
B.3.1 APICALL 114
B.4 Subroutine Listings 115
B.4.1 APIACTIV 115
B.4.2 APIDIUSB 117
B.4.3 APIDIUS2 122
B.4.4 APIDISOS 128
B.4.5 APIDIS2 130
B.4.6 APIFIL2 133
B.4.7 APIGTCMD 136
B.4.8 APILAST 137
B.4.9 APIPARSE 138
B.4.10 APIPTDOC 146
B.4.11 APIPURGE 148
B.4.12 APIRECVE 149
B.4.13 APIRTRVE 150
B.4.14 APISGNON 151
B.4.15 APISNOFF 153
B.4.16 APISUFIX 154

C.0 Sample EXECs for the CMS and PROFS User 155
C.1 Sample DBTMENU EXEC 155
C.2 Sample DBTSEND EXEC 157
C.3 Sample DBTLOGON EXEC 161
C.4 Sample DBTNOTE EXEC 163
C.5 Sample DBTRECV EXEC 164

LIST OF ILLUSTRATIONS

Figure 1. Interactive Communication between Box X and DISOSS . . . 7
Figure 2. Document Transfer from DISOSS to Box X 10
Figure 3. Document Transfer from Box X to DISOSS 13
Figure 4. DISOSS--Box X Interactive and Document Transfer Connections 17
Figure 5. DIU Structure Built by DBTSND1 30
Figure 6. Document Unit Segments for Document Text 30
Figure 7. Last Document Unit Segment 31
Figure 8. DIU Prefix 32
Figure 9. Request Distribution Command 32
Figure 10. Document Profile Information 33
Figure 11. Graphic Character Set Definitions 37
Figure 12. Transformations and Translations 39
Figure 13. PROFS menu to Add/Change a Document File 61

SYSTEM DESIGN

1.0 INTRODUCTION AND DESIGN OVERVIEW

The Document Interchange Architecture (DIA) defines a set of rules (or 'protocols'), allowing two programs to hold a 'conversation'; the purpose of this conversation is to exchange documents, together with commands describing what is to be done with these documents. DIA defines the 'language' the two programs use to exchange documents and control requests, but it is not concerned with the means by which these requests are moved from one program to the other. Typically, the transport function is provided by an SNA network; DIA does not duplicate any SNA function, and could theoretically use any transport mechanism that provides appropriate levels of reliability, recovery etc.

DISOSS exists to provide document handling functions, and to cooperate in this with intelligent devices such as 8100/DOSF, Displaywriter, 5520 and Scanmaster; hence it is logical that DISOSS 'converses' with these systems using exclusively the DIA protocols.[1]

DISOSS/370 V3R1 provides an Application Program Interface (API), which allows user-written CICS applications to use the DIA protocols to exchange documents and commands with DISOSS. In this case, since the user program and DISOSS are both executing under CICS, the transport mechanism for moving requests between them is not an SNA network but internal CICS and DISOSS facilities; however, the DIA protocols remain unchanged from those used to communicate between DISOSS and its other subsystems.

The API provides a means of interfacing almost any system to DISOSS in a manner that does not require modifications to the product, and should be safe from the effects of changes in future releases. Clearly, then, it is fundamental to any attempt to make DISOSS functions available to devices not directly supported.

The remainder of this chapter describes the basic characteristics of our proposed system design, and explains the logic that led us to that design rather than one of the many alternatives.

[1] Note that DISOSS does not communicate with the end-user, but with a program executing in the subsystem; it is the intelligent subsystem that provides end-user interfaces appropriate to the type of user it supports.

1.1 DESIGN OBJECTIVES

Our principal aim was to discover whether it was possible to design a system that embodied two fundamental characteristics:

- General Applicability
- Ease of Implementation

There was always the real possibility that these two characteristics would prove to be mutually exclusive, and so to make our aims more precise, we defined the following requirements of the design:

Generality

It should provide a generalised interface to DISOSS that could be used by the majority of systems installed in today's networks, including, but not limited to:

- VM/SP (and thus PROFS)
- S/34, S/36, S/38
- 8100/DPPX
- Series/1
- 5280
- PC

In addition, it must be structured so that an IBM customer could implement it without unreasonable difficulty or risk. This leads to the following three requirements.

Minimum New Code

It should make every possible use of existing products and functions, rather than demanding new ones specifically designed for the purpose.

Resilience

It must require no modifications to IBM-supported software.

Programming Skills

It should aim to avoid highly specialised or complex techniques and system interfaces, and should aim to use only those programming skills which are generally available in customer installations.

1.2 LOGIC OF THE DESIGN PROCESS

Given the objectives listed above, several major design decisions must be resolved at an early stage:

- Function Distribution. To what extent should we try to do things in the various subsystems rather than in the central CICS?
- User Interface. Should we attempt to define a standard one for all subsystems, or allow each to select its own?

- Available Tools. What standard system components and specialised program products could be utilised to avoid developing new code?

1.2.1 DISTRIBUTION OF FUNCTION

The subsystems under consideration vary widely in the programming facilities they offer: a VM system, for example, is vastly more powerful and flexible than a 5280, and a function that is straightforward to implement on one may be unreasonably difficult on the other. Furthermore, the range of programming skills needed to support several subsystem types may be very wide.

It therefore seemed sensible to minimise the amount of function performed in the subsystem, in order to minimise the amount of duplicated work in a network containing a variety of subsystem types. This is the most generalised approach, and is reflected in our design, but in a network containing only one subsystem type, it might well be worth reconsidering this decision.

1.2.2 END-USER INTERFACE

Each subsystem has its own interfaces to its end-users, and it would usually be desirable to retain some consistency with these; to achieve this, however, an installation with a variety of different subsystems may have to write the same function several times; on the other hand, each group of users could have an end-user interface that is consistent with the rest of the system.

Our conclusion was that the need to minimise duplicated effort would usually override the wish to provide tailored user interfaces, although, again, this balance might well change in a network containing a large number of one subsystem type.

1.2.3 AVAILABLE TOOLS

Two functions are common to all of the subsystems:

3270 Emulation The ability to appear to a host system as a cluster of remotely-attached 3270 displays. This may be a BSC or an SNA attachment, depending on the subsystem concerned, but both are supported by NCP and VTAM and can thus access any host application that supports 3270s.

Remote Job Entry The ability to emulate a batch terminal in order to send bulk data to the MVS host. The subsystems use a variety of protocols:

- BSC 2780 or 3780.
- BSC Multileaving Remote Job Entry (MRJE).
- SNA Single Logical Unit (SLU).
- SNA Multiple Logical Unit (MLU).

The first and third of these protocols are supported by CICS; all four are supported by JES.

Other communications facilities exist in some of the subsystems. For example:

- User programs in most subsystems may use SNA functions to communicate with a CICS application; however, the programming interfaces differ greatly, and the same programs could not be used in all subsystems.

- Some 3270 Emulation packages allow batch data to be read from disk and transmitted as if it had been keyed. This capability exists only on certain configurations of Displaywriter and PC, and there is no standard host software to perform the complementary receiving function for all implementations.

Since no communications function other than 3270 Emulation and RJE is common to all of the subsystems, our objective of generality requires that we base our system on them.

Given that all the subsystems can emulate 3270s, it is likely that DISOSS/PS will be a valuable tool, since its purpose is precisely to allow 3270 devices to make use of the DISOSS functions.

1.2.4 DISOSS/PS FOR INTERACTIVE FUNCTIONS

DISOSS/PS is a CICS application; it supports 3270 terminals, and provides end-user services for users of those terminals. Thus it can be regarded as an intelligent subsystem using DISOSS services, just as the DISOSS/8100, 5520 and Displaywriter subsystems do; the only difference is that DISOSS/PS executes under CICS, rather than in a separate machine, and communicates with DISOSS via the API rather than via an SNA session.

The components involved in connecting Box X to DISOSS via DISOSS/PS are shown in Figure 1 on page 7.

DISOSS/PS provides the 3270 user with access to most of the functions of DISOSS, including:

Distribution A mailbox for documents received from Displaywriter, DISOSS/8100, 5520 and other DISOSS/PS users. The following functions can be performed on a document in the DISOSS/PS mail-log:

- View
- File in the DISOSS Library
- Redistribute to other DISOSS users
- Print at the MVS host
- Delete

In addition, DISOSS/PS provides a limited text entry and editing function, so that simple documents can be created at the screen and filed in the library or distributed to other DISOSS users.

Library The DISOSS/PS user has access to the DISOSS document library and can use the following functions:

- Search
- View
- Distribute
- Print
- Delete

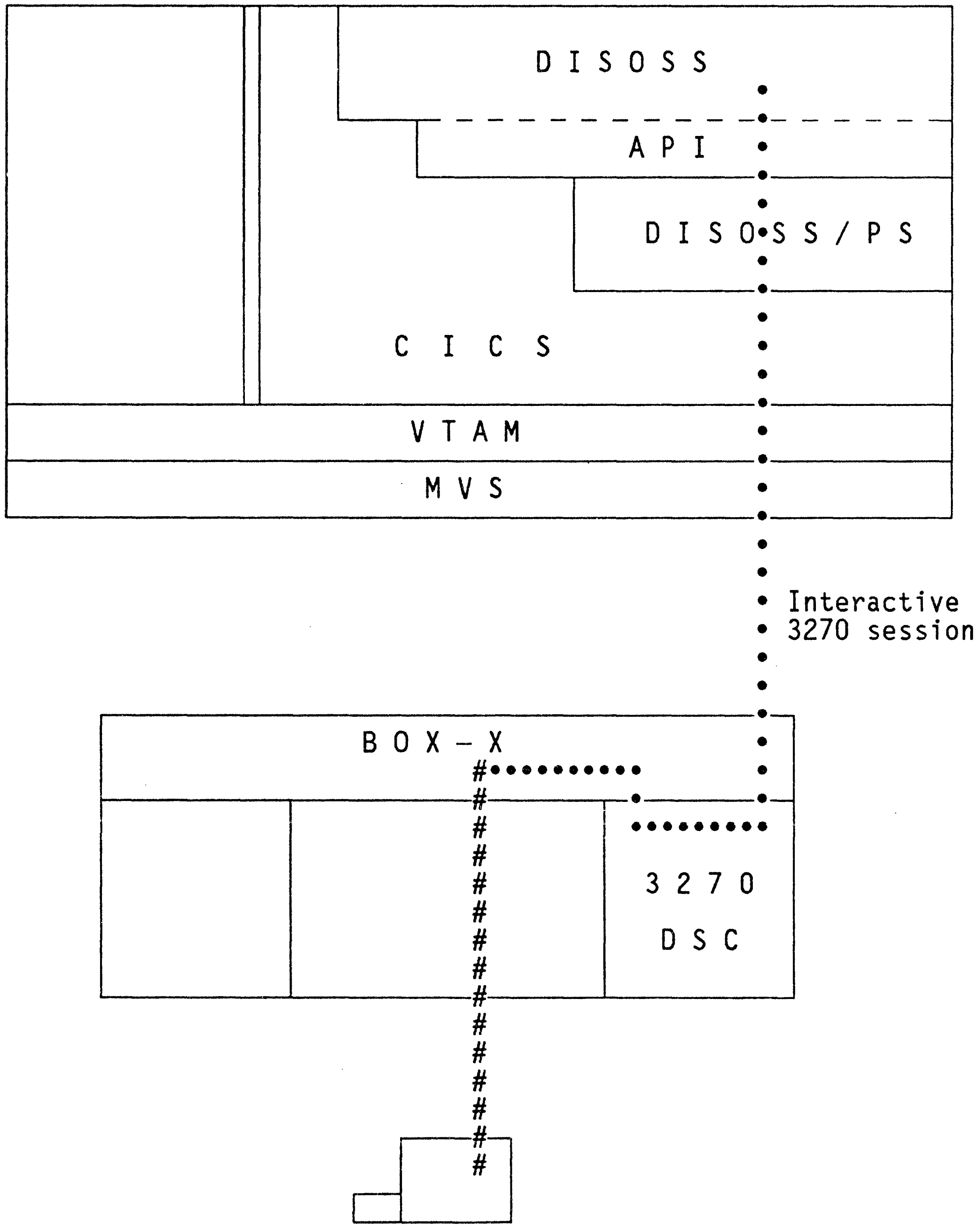


Figure 1. Interactive Communication between Box X and DISOSS

Clearly, then, DISOSS/PS provides many of the required functions to any subsystem that can emulate a 3270. However, precisely because it is designed for 3270 users, there are two important functions that it does not provide:

- A means of retrieving a document from the DISOSS/PS mail-log or the DISOSS library to the subsystem's own storage.
- A means of filing or distributing via DISOSS a document or file created and stored at the subsystem.

We need a means of moving documents in both directions between DISOSS and the subsystem; this is essentially batch data transfer, and our chosen vehicle for this is one of the RJE protocols.

1.2.5 RJE FOR BATCH FUNCTIONS

All of the subsystems can emulate a batch terminal of some kind, and CICS supports certain batch devices, so clearly it would be convenient to connect the subsystem to CICS and use a CICS transaction to send and receive files or documents. However, there are several difficulties:

- CICS does not support all of the protocols used by the various subsystems. In particular, it does not support the BSC MRJE and SNA MLU protocols.

- Using a BSC protocol to communicate with CICS would usually require a dedicated connection between Box X and CICS. This may imply a dedicated line plus two modems, which may be too expensive if the number of document transfers is low.[2] The SNA SLU protocol would avoid this problem, but is not supported by all subsystems.

- Not all subsystems can support multiple physical connections to a host, and if one is already installed, it is more likely to be communicating with an RJE system than with CICS.

- Subsystems designed in the expectation of communicating with an RJE system may be more difficult to operate when communicating with a user-written CICS transaction.

- JES has good facilities for recovering from communications failures, safe-storing and re-transmitting data. Equivalent function, if desired in a CICS connection, would have to be programmed by the user.

- JES is normally available whenever the operating system is running; thus a job can be submitted whenever the network is available, regardless of whether CICS is executing or not.

For these reasons, we concluded that the most general approach would be to have the subsystem communicate with the RJE system rather than directly with CICS. This leads to the question of how documents are to be moved in both directions between JES and CICS.

[2] It may be possible to reduce this cost in some cases:

- Channelised modems would allow two or more 'separate' connections to use the same physical link. This would still require separate 37x5 ports for each connection, and depends on the total traffic being low enough to allow the bandwidth to be divided up in this static way.

- Use of the Non-SNA Interconnection (NSI) licensed program in the 37x5 would allow the subsystem to switch its connection from one host application to another, for example between JES and CICS. This, of course, implies that only one communication function could use the physical link at a time, which may be operationally unsatisfactory.

1.2.6 MOVING DOCUMENTS FROM CICS TO JES

We could write a CICS transaction to obtain documents via the DISOSS API, transform them into a datastream appropriate to Box X, then submit batch jobs (via the operating system's internal reader facility) to place the transformed documents on the JES spool.

As it happens, however, there is a standard function in DISOSS which will achieve the same result. The Host Print facility submits a batch job whose function is to transform a document from its DISOSS form into a series of print-lines appropriate to a host-attached 1403 printer. This print data is output to the JES spool, from where it can be printed on a real printer, or routed to some other destination known to JES. This destination could of course be our remote subsystem.

Thus, the DISOSS Host Print function, together with appropriate JCL statements to direct its output, provides the means of moving a document from DISOSS to Box X, without the need for any user programming. Figure 2 on page 10 shows the connection involved.

1.2.7 MOVING DOCUMENTS FROM JES TO CICS

There is no standard DISOSS function to meet this need, so this is a more difficult problem; nevertheless, there are several possible solutions:

Direct SNA Session between JES and CICS, or between the submitted batch job and CICS.

SNA Relay Program receiving data from JES on one side, and passing it on to CICS on the other.

JES External Writer executing under CICS and reading data from the JES spool.

Shared Dataset allowing the batch job to insert data, and CICS to retrieve it.

1.2.7.1 Direct SNA Session

The Network Job Entry (NJE) functions of JES2 and VSE/POWER (but not JES3 or OS/VS1 RES) allow one JES system to pass jobs and output across an SNA session to another JES system. If such a session could exist between JES and CICS, then JCL statements in the job submitted by Box X could cause the job to be passed on to CICS rather than executed by JES. Unfortunately, however, such a session is not possible since CICS does not support the particular set of SNA protocols required by JES for this purpose.

Alternatively, the job submitted by Box X and executed by JES could certainly establish an SNA session with CICS and pass the document across this session to a user-written CICS transaction. The main problem is one of recoverability. If the program cannot establish its session with CICS for some reason, it cannot continue; but if it terminates, its input data is lost, and the originating user must resubmit his request. It would be preferable to ensure that the data

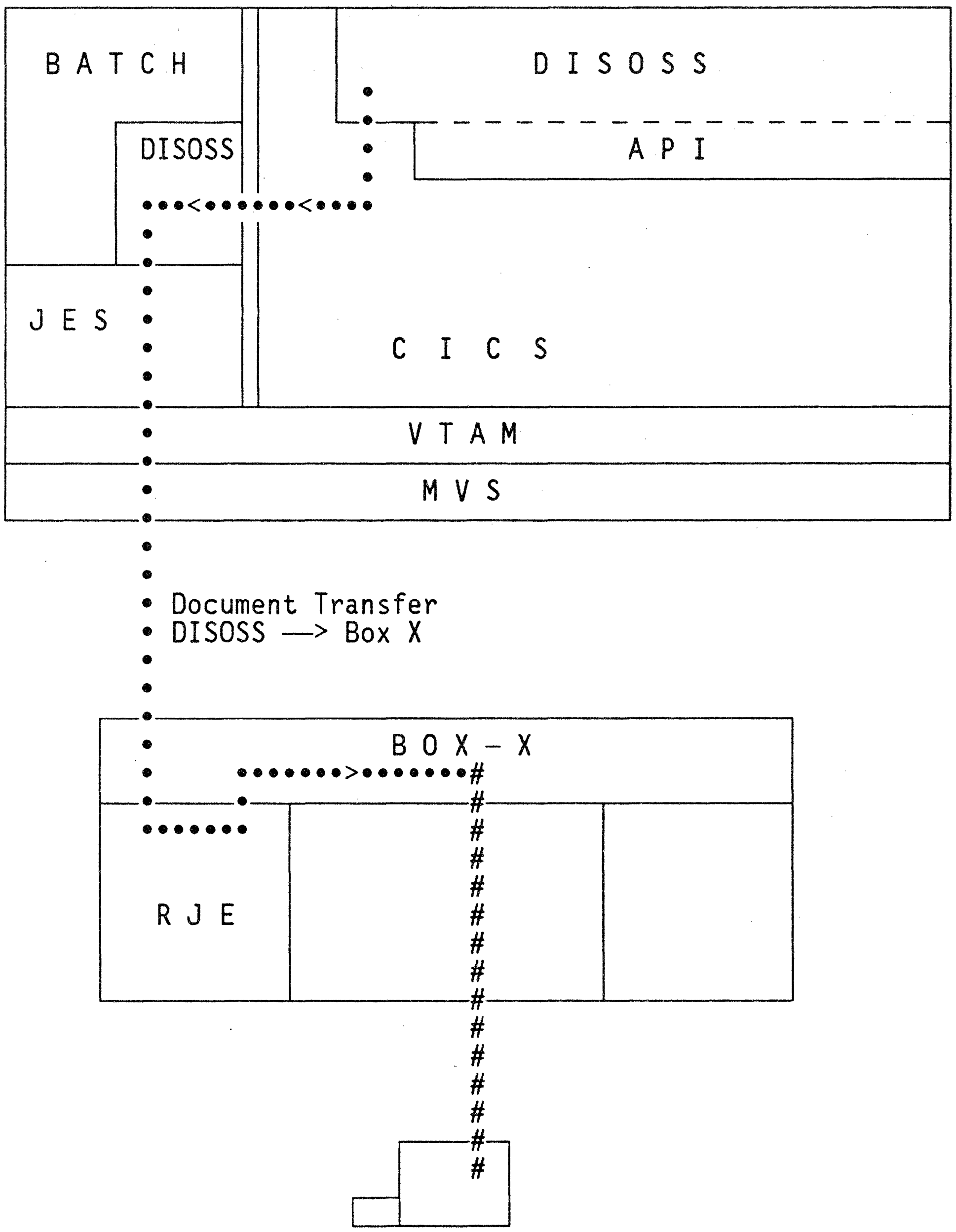


Figure 2. Document Transfer from DISOSS to Box X: this assumes an SNA connection. If the RJE connection were BSC, RTAM would be used instead of VTAM.

remains safely on the JES spool until we know it can be delivered to CICS, and this cannot easily be achieved with this approach.

1.2.7.2 SNA Relay Program

This program would be a long-running task, active in the system whenever JES and CICS are running. It would establish two SNA sessions:

1. With CICS. This might be one of several session types supported by CICS; the most convenient would probably be either the SNA SLU batch session, or the LU6.2 session which is specifically designed for program-to-program communication.

2. With JES. Only when the session with CICS is active would this second session be started, thus avoiding the recoverability problem noted earlier. This session might use the SNA SLU protocol to appear to JES as a 3770 RJE device. Alternatively, it might use the SNA Network Job Entry protocol, so that it would appear to JES as another JES. In this case, the batch job submitted by Box X would not be executed by JES, but would be passed to the relay program, which would in turn pass it on to CICS.

This approach avoids the problems of both types of direct session discussed above: no batch jobs need be executed by JES, yet the JES spool is used as a safe store until the data can be delivered to CICS.

The disadvantage is that the VTAM programming skills needed to implement it are not universally available, and for that reason it must be rejected in our case.

1.2.7.3 JES External Writer

An external writer is a user-written program using interfaces provided by the operating system to read output data directly from the JES spool. If such a program were executing as a CICS application, it could read JES output into the CICS system with no need for SNA sessions. Unfortunately, the interfaces provided by MVS and VSE are not suitable for use under CICS, since they issue WAIT macros which would cause the entire CICS system to wait; the alternative is to implement the External Writer via a user SVC, but again we felt that this type of programming skill would not be widely available.

A further disadvantage is that this interface can only handle SYSOUT data: in other words, the incoming data is always 1403 printlines. This will often be satisfactory, but cannot handle the possibility that Box X may wish to send some more sophisticated datastream.

1.2.7.4 Shared Dataset

An apparently very simple approach to the problem is to allow the job submitted by Box X to write its document into a disk dataset, from where it can subsequently be read by a CICS transaction. There are still some potential difficulties, however:

- The dataset must be concurrently shared between CICS and the batch job, since it would not be satisfactory to run the batch jobs only when CICS was down. JCL can allow both CICS and the batch job to allocate the dataset, but to avoid data integrity problems we must ensure that concurrent updating is not allowed, or is very carefully controlled. VSAM can simply overcome this problem, by ensuring that only one task can open the dataset at a time; alternatively, concurrent updating can be permitted with VSAM, at the cost of increased user programming effort.

- If the dataset is to be used concurrently by batch and CICS over a long period, then we must be able to insert and delete records; this in addition to the need for frequent OPEN and CLOSE functions from both batch and CICS, could create a significant performance bottleneck if not considered carefully in the system design.

However, this approach has the major advantages of being easy to program and maintain, since VSAM file operations are well understood in most installations. Also, having examined the potential performance problems, we felt that careful dataset and application design could contain them to acceptable levels. We therefore concluded that, given our original objectives, this technique would be most appropriate. Figure 3 on page 13 summarises the connection for document transfer from Box X to DISOSS.

1.2.8 CICS APPLICATION TO ACCESS DISOSS API

By using the DISOSS Host Print function plus NJE for document transfer from DISOSS to Box X, we eliminate the need for user-written CICS programs to obtain documents via the API and redistribute them to Box X users; thus new code is only needed to pass documents into DISOSS from Box X. The next question is, how much DIA function needs to be supported in this new user code? Any DIA application must support the SIGN_ON, ACKNOWLEDGE and SIGN_OFF commands, and in this case we will also need either FILE or REQUEST_DISTRIBUTION (or both) in order to pass documents into DISOSS:

- **FILE** requests that DISOSS file the attached document in its library. This command has many operands, since users may want to specify many attributes of a document in order to ensure that it can later be retrieved in a satisfactory way. Any of the following may be specified, for example:

  - Document Name
  - Author
  - Subject Matter
  - Recipients
  - Keywords for later search operations
  - Document Class (e.g. Memo, Letter, Report etc.)
  - Access Codes to ensure the desired level of security

  Consequently, of course, 'File' is potentially a complex command, and therefore a variety of end-user errors could occur and would need to be handled by our code. For example, if the end-user specified an invalid Access Code,

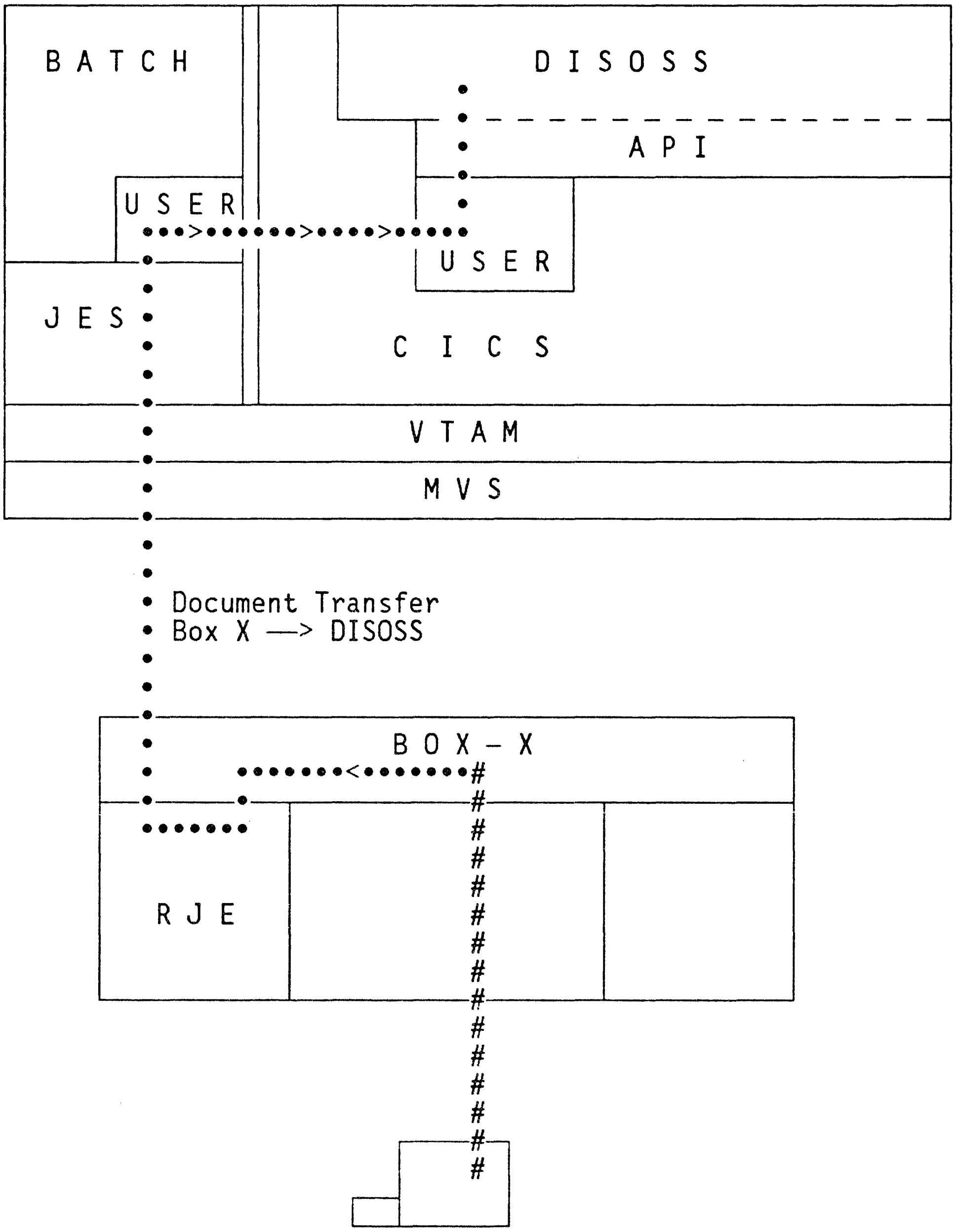


Figure 3. Document Transfer from Box X to DISOSS: the batch user program communicates with the CICS user program via a shared VSAM dataset not shown in this diagram.

DISOSS could not perform the File operation and would inform us via the API. We would either have to discard the document and notify the end-user, or ask the end-user to correct his request so that we could retry the File command.

- **REQUEST_DISTRIBUTION** asks that DISOSS distribute the attached document to a named user or users. Since the document is not permanently stored in the library, there is usually no need for all the descriptive information required by the File command. Often, only the following would be needed:

- Document Name
- Recipients

However it would probably be desirable for the end-user to have, in addition, at least the following:

- Distribution Lists, so that he could simply name a list in order to have a document distributed to several users.
- Priority Distribution, so that he could designate one document as more urgent than another.
- Personal Distribution, so that he could designate a document as Personal to the recipient (i.e. not available to the recipient's secretary).

These functions, of course, would also add complexity both to the Box X end-user interface, and to the user-written CICS programs.

A reasonable interface should allow the Box X user to use both the filing and distribution services of DISOSS, but a serious difficulty arises: there is no interactive communication between the Box X user and the CICS application, since document transfer occurs via batch RJE facilities, and thus there is no reasonable way to converse with the end-user when DISOSS or our code detects an error in his request. All we can do is send him a report identifying the error and asking him to resubmit his request at some time in the future. Obviously this is not ideal, but the more function we offer the end-user, the greater the risk of such errors occurring.

Thus we have conflicting requirements:

- Full support of the File and Request_Distribution functions in the user API programs will:

  - Add complexity to the end-user interface at Box X, to allow the user to specify the various options.
  - Add complexity to the user API programs to handle these options, build the more complex DIA commands, interpret the possible error notifications returned by DISOSS, and return some useful message to the end-user.
  - Still only offer a batch-type interaction with the end-user, which he is unlikely to find attractive.

  This conflicts with our objectives in two ways:

  1. It will require significant user code in each subsystem type to support the more complex end-user interface.
  2. It adds considerable complexity to the CICS application, and therefore threatens the ease of implementation objective.

- On the other hand, the requirement that our system be generally applicable across a wide range of subsystems and user groups, demands that the full range of the DISOSS Library and Distribution services be made available.

The solution to this problem was surprisingly simple:

1. The Box X user is already a user of DISOSS/PS, and DISOSS/PS already provides extensive support for the full range of filing and distribution functions. Furthermore, it does so interactively, giving the end-user a chance to correct an invalid request and resubmit it at once.

2. Our CICS application can distribute the Box X document to the DISOSS/PS userid representing the Box X end-user. Thus the Box X user will see the document in his mail-log, and can use all the facilities of DISOSS/PS to file or distribute it.

3. The user-written CICS code is thus greatly simplified, since it only has to support a simple form of the Request_Distribution command[3], with a much reduced likelihood of errors. Furthermore, the subsystem-unique code is also simplified, since the complexities of the end-user interface for full support of filing and distribution are now handled by DISOSS/PS.

This approach views the user-written code as just a means of delivering Box X documents to DISOSS/PS, which is the principal means for the Box X user to interface with DISOSS. Apart from minimising the complexity (and thus the maintenance) of the user-written code, it also increases the likelihood that the Box X user will be able to take advantage of any new function in DISOSS or DISOSS/PS, without being dependent on corresponding enhancements in the user-written CICS programs.

[3] In fact it may also be desirable to support a simple form of the File command. This is because a document delivered to DISOSS/PS is transformed on receipt into a simple DISOSS/PS internal datastream; it is not always possible to reconstruct the original datastream when the document is subsequently filed or redistributed from the DISOSS/PS mail-log. In such cases, our system can perform a simple File operation on behalf of the DISOSS/PS user, who thus becomes the owner of the document in the DISOSS library, and can therefore update the profile or redistribute the document as he wishes.

1.3 OVERVIEW OF THE SYSTEM STRUCTURE

Figure 4 on page 17 illustrates the general organisation of our system design, intended to allow the Box X user to work interactively with DISOSS via DISOSS/PS, and to transfer documents in both directions between DISOSS and Box X.

To summarise the main characteristics:

1. DISOSS/PS is the principal end-user interface to DISOSS, and is used for all interactive functions.

2. DISOSS/PS invokes the DISOSS Host Print function in order to move a document to Box X from either the DISOSS library or the DISOSS/PS mail-log.

3. RJE and user-written batch and CICS programs are used to move a document from Box X to the Box X user's DISOSS/PS mail-log, from where the full functions of DISOSS/PS are used to file or distribute it.

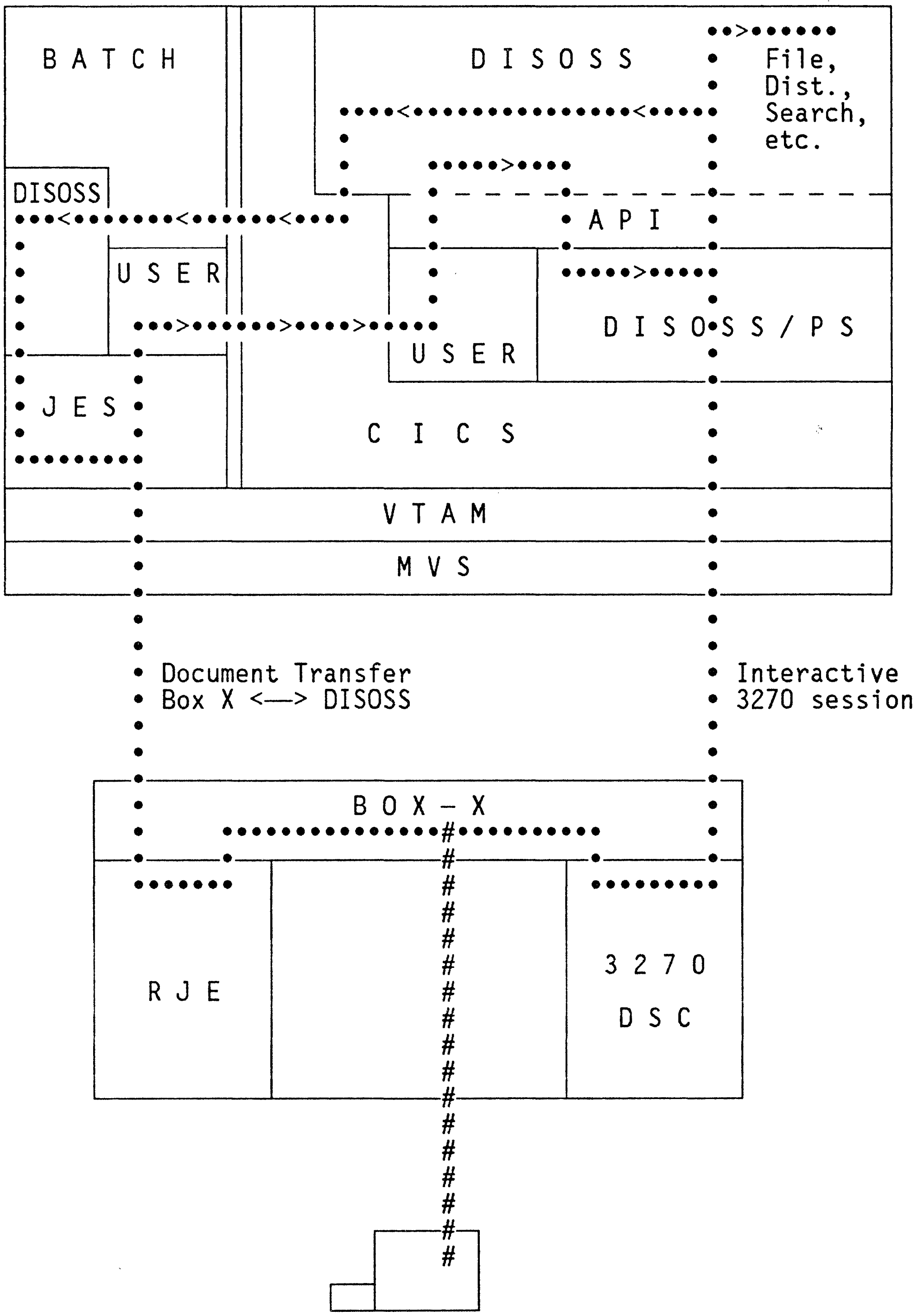


Figure 4. DISOSS--Box X Interactive and Document Transfer Connections

2.0 SYSTEM STRUCTURE

This chapter describes the general structure of the suggested design, and summarises the functions of the various components.

"Introduction and Design Overview" on page 3 has shown that our system has three logical components:

- Interactive communication between the Box X end-user and DISOSS/PS.

- Document transfer from DISOSS to Box X.

- Document transfer from Box X to DISOSS.

2.1 INTERACTIVE COMMUNICATION

This is implemented by a combination of existing product functions, and all that is required to deliver the needed end-user function is appropriate customising of the following:

- Box X 3270 emulation component.
- NCP, VTAM and CICS definitions for the emulated 3270 terminals.
- DISOSS/PS definitions for the Box X end-users.
- DISOSS Host User Profile definitions for the DISOSS/PS users.

The general organisation is shown in Figure 1 on page 7, and this component of the system is not further described in this chapter.

2.2 DOCUMENT TRANSFER FROM DISOSS TO BOX X

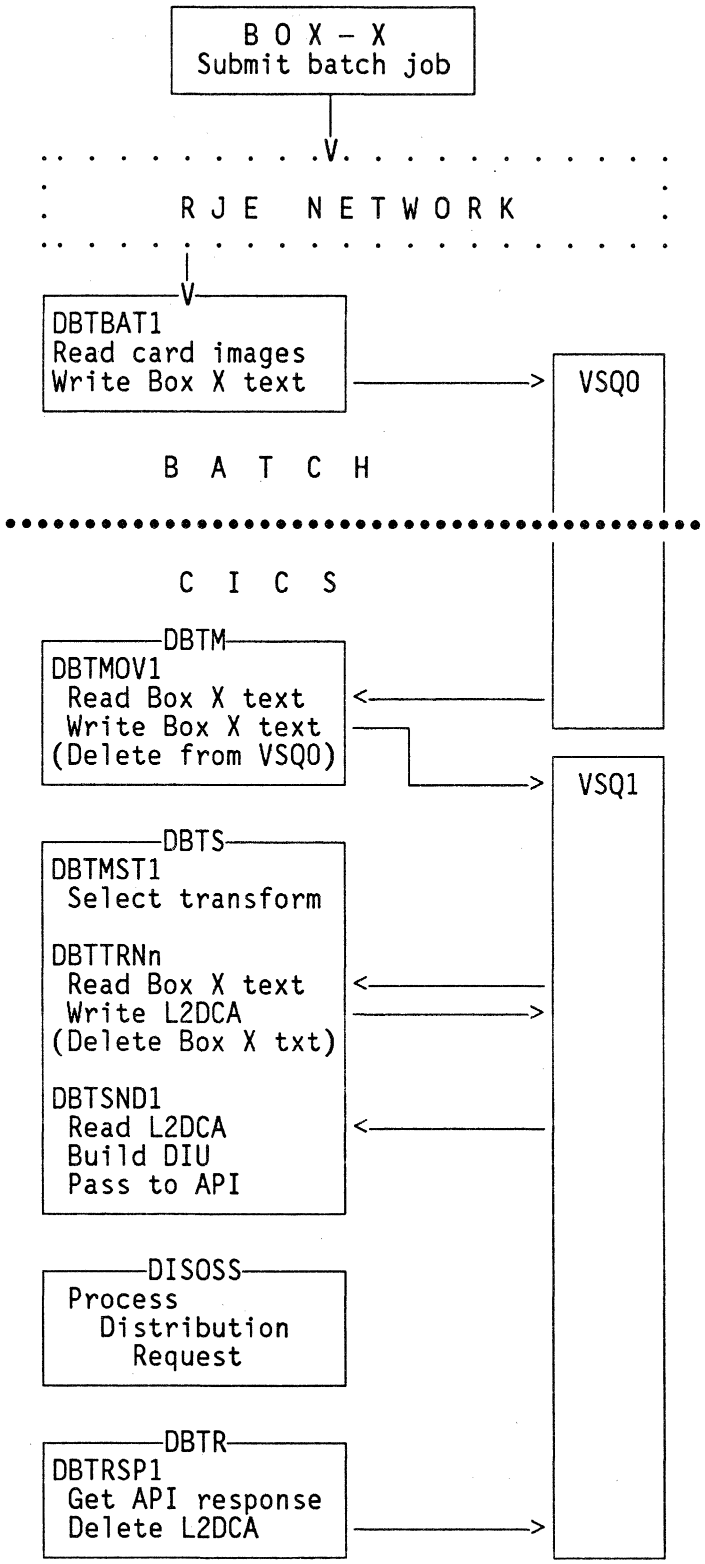
This too is implemented by a combination of existing product functions, and all that is needed is appropriate customising of the following:

- Box X RJE component.
- Definitions for the Box X RJE component in JES and in EP or NCP/VTAM.
- JCL procedures to route output to Box X.
- DISOSS Printer Description Table definitions to generate the appropriate Host Print batch jobs.

The general organisation is shown in Figure 2 on page 10, and this component of the system is not further described in this chapter.

2.3 DOCUMENT TRANSFER FROM BOX X TO DISOSS

This component of our system is the only one requiring user-written code, and will be the subject of the remainder of this chapter. Figure 5 on page 20 shows the major components involved.


BOX-X
Submit batch job
RJE NETWORK
DBTBAT1
Read card images
Write Box X text
VSQ0
BATCH
CICS
DBTM
DBTMOV1
Read Box X text
Write Box X text
(Delete from VSQ0)
VSQ1
DBTS
DBTMST1
Select transform
DBTTRNn
Read Box X text
Write L2DCA
(Delete Box X txt)
DBTSND1
Read L2DCA
Build DIU
Pass to API
DISOSS
Process
Distribution
Request
DBTR
DBTRSP1
Get API response
Delete L2DCA

2.3.1 MAJOR COMPONENTS

2.3.1.1 Box X Job Submission

User programming in Box X performs the following functions:

- Constructs a document header record to identify the document, its format and its destination, so that the appropriate DIA request can be constructed at the host.
- Breaks the text lines into 80-byte card images.
- Adds JCL to invoke the appropriate user-written batch program, and submits the job via the RJE system.

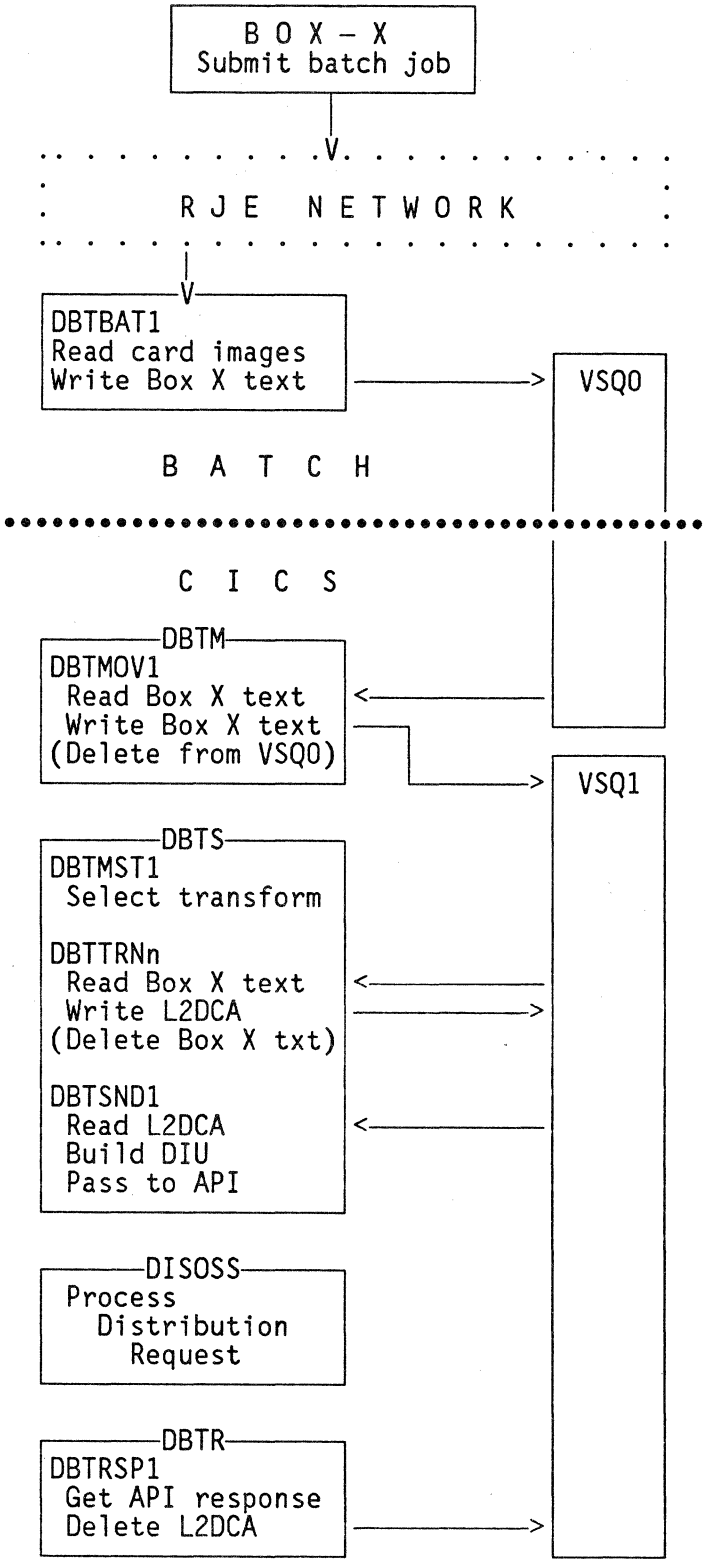
2.3.1.2 Batch Program

DBTBAT1 is a user-written batch program whose purpose is to insert the Box X document in the shared VSAM dataset DBTVSQ0. In order to minimise contention for the dataset, and to minimise the number of insert and delete operations needed to transfer the document, DBTBAT1 does the following:

- Builds a document header from the header card sent by Box X.
- Reads the card images sent by Box X and rebuilds the original text lines.
- Concatenates these lines of text, separated by X'1E' Interchange Record Separator characters, in large physical records.
- Adds a unique document identifier to be used later as part of the VSAM key, then writes the large records to a temporary dataset.
- When the document is complete, opens the the shared VSAM dataset and inserts the contents of the temporary dataset.

2.3.1.3 CICS Program DBTSON1

This program starts a DIA session with DISOSS via the API, and would normally be executed at CICS start-up time. Having established the DIA session, it could then initiate transaction DBTM to start document transfer.


BOX-X
Submit batch job
RJE NETWORK
DBTBAT1
Read card images
Write Box X text
VSQ0
BATCH
CICS
DBTM
DBTMOV1
Read Box X text
Write Box X text
(Delete from VSQ0)
VSQ1
DBTS
DBTMST1
Select transform
DBTTRNn
Read Box X text
Write L2DCA
(Delete Box X txt)
DBTSND1
Read L2DCA
Build DIU
Pass to API
DISOSS
Process
Distribution
Request
DBTR
DBTRSP1
Get API response
Delete L2DCA

2.3.1.4 CICS Transaction DBTM

This transaction consists of one program, DBTMOV1, whose purpose is to move Box X documents from the shared dataset DBTVSQ0 to an identical VSAM dataset, DBTVSQ1, which is wholly controlled by CICS. Functions are:

- Call subroutine DBTOPN1 to open the shared dataset.

- If the open is unsuccessful, schedule a new DBTM for a later time, and end.

- If the open is successful, copy all DBTVSQ0 data into DBTVSQ1.

- For each complete document transferred, initiate transaction DBTS, passing it the key of the document in DBTVSQ1, and delete the document from DBTVSQ0.

- Call subroutine DBTCLS1 to close the shared dataset.

- Before ending, schedule a new DBTM transaction to execute a few minutes later.

2.3.1.5 CICS Transaction DBTS

This transaction contains the main processing of the system, and consists of several programs:

- Program DBTMST1 examines the document header in DBTVSQ1, identifies an appropriate program to transform the input datastream to the desired output datastream, and invokes that transform routine.

- The chosen transform program, DBTTRNn, converts the text of the DBTVSQ1 distribute request to a datastream understood in the DISOSS system (probably the DCA Level 2 Final Form Text datastream), and passes control to program DBTSND1.

- DBTSND1 builds the appropriate DIA structure, called a Document Interchange Unit (DIU), which contains a DIA Request_Distribution command and the document text, and passes it across the API to DISOSS, requesting that transaction DBTR be invoked to process the subsequent response from DISOSS.

2.3.1.6 CICS Transaction DBTR

This transaction consists of one program, DBTRSP1, which checks that the distribution request was successful, and deletes the document from DBTVSQ1.

SAMPLE IMPLEMENTATION

3.0 SAMPLE SYSTEM IMPLEMENTATION

This chapter describes the sample implementation that was made to validate the system design. We used VM/CMS as our Box X, and therefore this sample has only implemented support for one inbound datastream, the 1403 printline.

It is important to remember that the purpose of this implementation was solely to show that the proposed design was workable. It was never our intention to produce an implementation that could be used in a production environment: that would have required more detailed documentation, more extensive error detection and recovery code, and more rigorous programming and testing techniques than were possible in the timescale of our project.

This sample implementation is only intended as a demonstration, and is not appropriate for any other use.

3.1 MOVING DOCUMENTS FROM DISOSS TO BOX X

The design of a CICS-Batch interface can be straightforward, because CICS applications can write jobs to the internal reader. The DISOSS Host Print facility uses this technique, so rather than duplicate many of the functions of DISOSS, the Host Print facility was used as the basis for the DISOSS to Box X interface.

The DISOSS Host Print facility works as follows:

- The DISOSS user enters the name of a logical printer in the Destination Name field of a Host Print menu.
- A batch job is written to the internal reader for execution. The batch job contains the following:
  - A jobname derived from the Host Print Menu jobname field.
  - Jobcard parameters derived from:
    - The accounting information in the Host User Profile of the DISOSS user.
    - The JOBJCL option that was specified in the Host Definition job during installation
  - An invocation of the format procedure whose name was specified in the JOB option of the PDT entry for the logical printer entered in the destination name field of the Host Print menu.
  - An input dataset with the document text inline.
- A format program, usually the DISOSS supplied DSVOL500, should do the following:
  - The document is formatted into 1403 printlines, according to the format options and the printer characteristics data specified in the relevant entry in the Printer Description Table (PDT).
  - The print/fidelity table index DSVS5800 is searched for a table with an input GPID and output GCID corresponding to the document GPID and the printer GCID specified in the PDT entry. The output characters are translated if a table is found.
- The output document is routed by JES to the printer determined by the procedure JCL.

3.2 MOVING DOCUMENTS FROM BOX X TO DISOSS

3.2.1 USE OF THE DISOSS API

3.2.1.1 General Remarks

One of our design objectives was to avoid the need for specialised programming skills wherever possible; thus we have rejected designs requiring the use of VTAM programming or the JES External Writer interfaces, and the DISOSS API itself is the only requirement for special programming knowledge. We do not attempt to describe the API in this book, and recommend that the following documents be regarded as essential reading for anyone wishing to understand this implementation in detail:

SC30-3096 DISOSS/370 Version 3 Application Programming
GG24-1614 DISOSS Application Interface: Programming Guidelines

However, for the general reader, the following points may be helpful.

- The API consists of a queue (implemented as a VSAM KSDS), and a set of commands to insert and retrieve data on that queue. The data itself must be in the form of a DIA-defined DIU.
- A DISOSS supplied module (DSVAW000) must be part of the user transaction to invoke the API commands.
- Every transaction that wants to use the API has to issue an API 'Activate' command first, and provide a DISOSS user name for the session. Then, the first command on the DIA session must be an API-BIND, which will build a DIA Sign_On.
- Multiple DIUs can be put on this queue by different CICS transactions for the same DIA session. On an 'Activate' command, the API takes a unique time stamp for that transaction and username of the DIA session. This time stamp will be used as a key field for all data of this transaction put on the API-queue.
- DIUs will only be processed after receipt of a API 'Last' command, and a syncpoint of the user transaction.
- Only one DIU will be processed for every 'Last' command.
- Updates on the API queue will be backed out if an abend occurs.
- The responses for DIA commands that are processed through the API must be analysed by a new transaction named in the API 'Last' command. That is, the API is an asynchronous interface.
- The API gives the 16 byte DIU-ID field as part of the receive data for the response. (DIU-ID is provided by the originator of a DIU).

3.2.1.2 Our Use of the API.

The Box X to DISOSS transactions distribute documents into DISOSS by means of a 2-way multithread communication with the DISOSS API; that is, multiple DBTS transactions may concurrently pass documents into the API, and multiple DBTR transactions may concurrently receive results, but all share the same DIA session with DISOSS and are regarded by DISOSS as one user.

Sign_On and Sign_Off are provided by separate CICS transactions. Transaction DBTN may be used to perform Sign_On if the CICS start-up procedure does not invoke program DBTSON1 to do so. DBTF performs a DIA Sign_Off, though this is not normally required.

Transaction DBTS sends a distribute request to the API. For each document it builds one DIU of the structure shown in Figure 5.

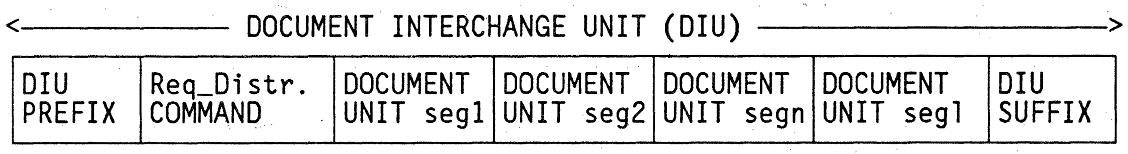


Figure 5. DIU Structure Built by DBTSND1: the maximum number of document units in one DIU is 255 (DIA architecture limitation). There is no limit on the number of segments in a document unit.

The structure of document unit segments that contain L2DCA data is shown in Figure 6. As there is no limitation on the number of segments in one document unit, there is also no limit on the size of a document to be passed through the API (except, of course, the size of the API-queue dataset).

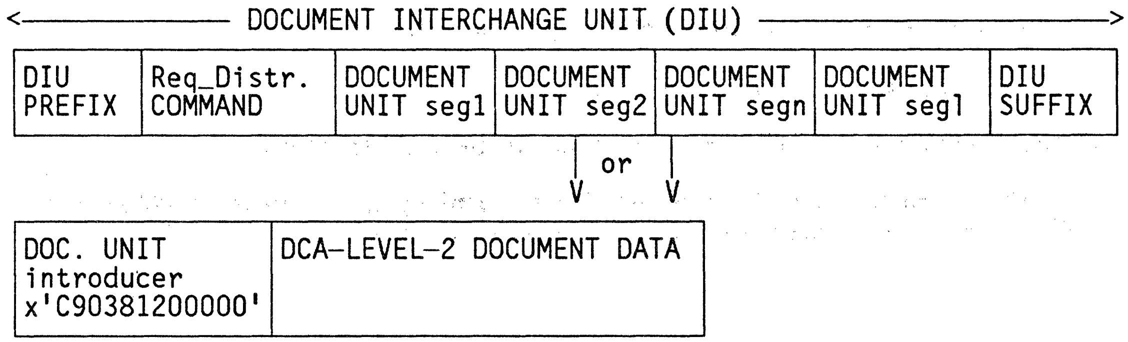


Figure 6. Document Unit Segments for Document Text: these are middle-in-chain segments, with only L2DCA data.

The document unit is ended by a segment with only a 'last segment indicator' and no data, shown in Figure 7.

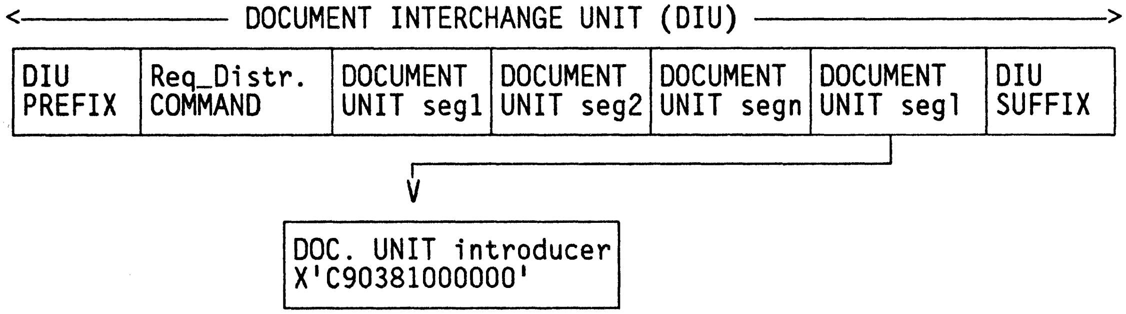


Figure 7. Last Document Unit Segment: there is no text content, only a last segment indicator.

These DIA structures are built by the DBTSND1 program, which calls a set of subroutines to perform particular functions:

- APIACTIV issues the API 'ACTIVATE' command for user DIST01, which is the API user solely defined for this interface in the DISOSS Host User Profile dataset.
- APIGTCMD issues API 'RECEIVE' to receive data from the API queue. It parses the data and sets return code fields for the caller.
- APIDIS2 (or APIFIL2) builds the first DIU part, up to and including the first document unit containing the document base profile, and issues API 'SEND' to pass it to the API.
- APIPTDOC sends the remainder of document unit segments through the API.
- APISUFIX passes the DIU suffix through the API.
- APILAST issues API 'LAST' command.
- APIPURGE issues API 'PURGE' command in case of errors.

Refer to **DISOSS Application Interface: Programming Guidelines, GG24-1614** for details of the design and operation of these subroutines.

DIU Correlation Data

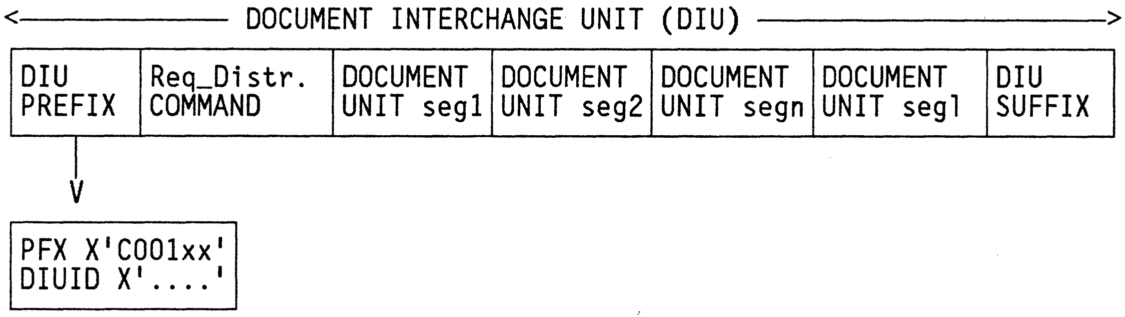


Figure 8. DIU Prefix: the DIU-id field can be expanded to 16 bytes, and is used in this system to correlate responses in DBTR with requests from DBTS.

The first 16 bytes of the DBTVSQ1 record key are used as the correlation data in the DIU prefix. This data is unique, being made up of the date and time of creation, together with an identifier of the creating interface (which in this case is always the DBTBAT1 program). It is used as the document identifier in the API communication, and it is passed to the response transaction DBTR. It can be used by DBTR to identify the processed Box X request on DBTVSQ1, so that appropriate action can be taken. In the current design, this action is to delete the request from DBTVSQ1 if it was successful, and to issue an API PURGE if it was unsuccessful.

Request_Distribution Command

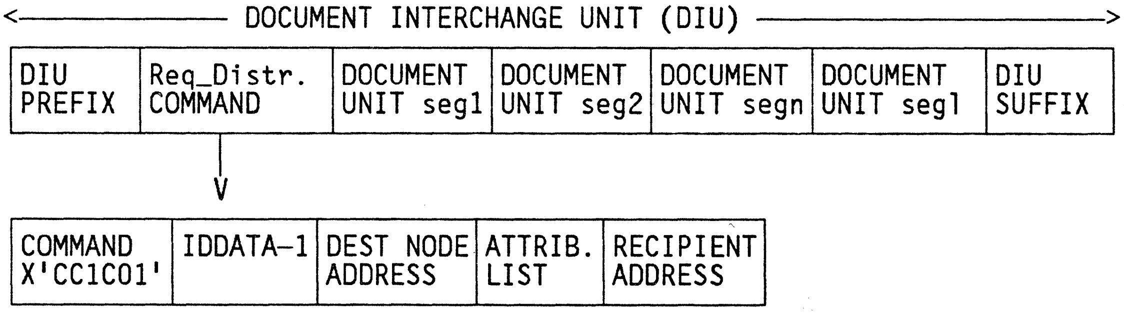


Figure 9. Request Distribution Command: attribute list and recipient address are required parameters.

The command itself is in the ARR (asynchronous reply required) command class designated by X'CC' in the 'I' byte of the command. ARR commands will be replied to with an ACKNOWLEDGE command together with correlation data. It contains the following parameters:

- Identified Data format 1, for pointing to the first document unit in this DIU as the document for this command.
- Destination node address format 1, an 8 byte parameter designating the office systems node for the recipient address.
- Attribute list format 1, a required parameter, specifying:
  - no confirmation of delivery,
  - not personal,
  - no priority,
  - number of copies = 1.

  DIA gives the option to specify a 256 byte message in this parameter, but this is not used by our programs.
- Recipient address format 1, a required parameter up to 8 bytes long.

The address combination would usually be a DISOSS/PS user because DISOSS/PS provides all necessary functions for documents in the mail-log; however, a simple distribution to one user can still be done directly.

Document Profile

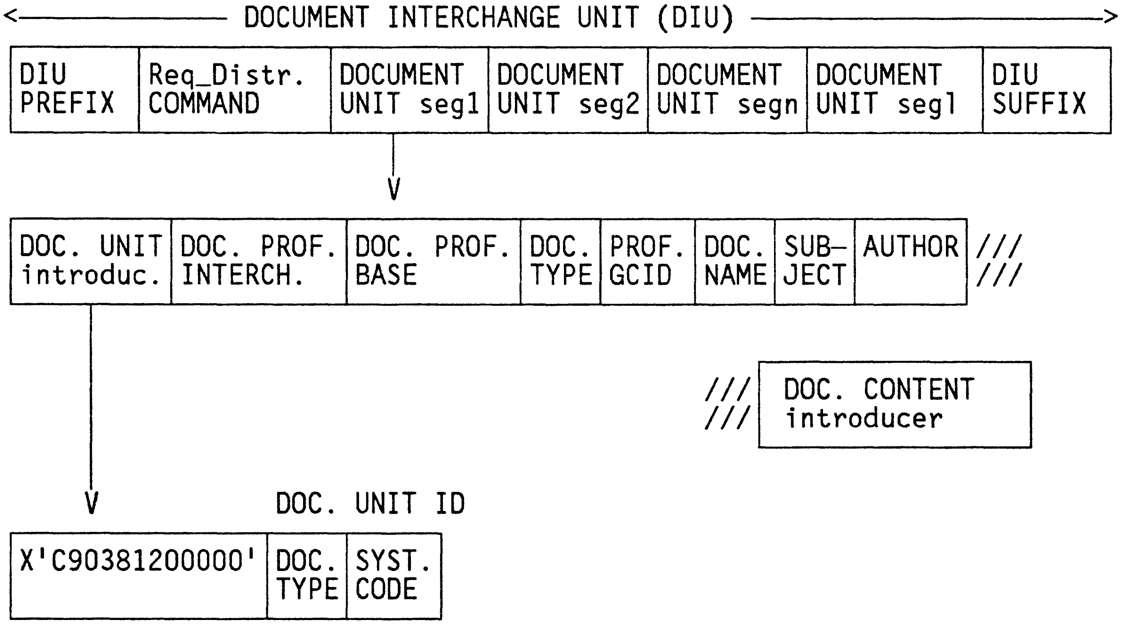


Figure 10. Document Profile Information: the first document unit segment contains the document profile only. The first document unit introducer in a segmented chain has a document unit id.

The first document unit contains the document profile. We provide the following parameters:

- Document name, a required parameter, maximum length of 15 characters in our design (44 characters DIA maximum). According to DIA rules the first and last character of the name may not be a space.

- Document type, a required parameter, a 2 byte field in our case always X'0002' for L2DCA data.

- Profile GCID, a required parameter, in our case always set to X'01510100'. This is the standard GCID in DISOSS for profile data.

- Document GCID is omitted from the document profile. The document GCID is set in the DCA level 2 datastream to X'00D70108'. This is the standard GCID in DISOSS for the 1403 TN chain. DISOSS will look in the DCA datastream for the document GCID. See "Datastream Transformations" for a discussion of why this GCID was chosen.

- Subject, always set to 'Mailbox Project   '.

- Author, always set to 'Mailbox   '.

3.2.2 DOCUMENT TRANSFORMATIONS AND TRANSLATIONS

3.2.2.1 Datastream Transformations

From Box X inward the following data transformations are performed on 1403 print lines; a header record identifies the incoming data as 1403 rather than any other datastream.

1. In Box X: reformatting of print lines to 80 byte cards for transportation through the RJE system as SYSIN data.

   Certain boxes might have abilities to transport data in a more efficient way through a network. However the design of these programs would then no longer be general.

2. In the DBTBAT1 program: re-blocking of 80 byte SYSIN data to print-lines and blocking of print-lines in records with a maximum size of 5959 bytes (+ key of 39 bytes + 2 bytes length field = 6000 bytes). All trailing blanks in the print lines are deleted. Blocking and deletion of trailing blanks is done solely for efficiency purposes. For unblocking purposes the following additional data transformations are done:

   - Print lines are separated by IRS codes, X'1E'.

   - Except for the first byte, which should be the 1403 print control character, all characters below X'40' are converted to X'40', to ensure that no extraneous X'1E' characters will be present in the print data. (With DCF, any character could have been generated as print data).

     This serves an additional purpose: In L2DCA, multibyte and one-byte controls are used to control the final printing of data; all multibyte controls start with X'2B', so the above transformation ensures that no unwanted multibyte controls are present in the 1403 datastream. Single byte controls above X'40' in DCA Level 2 that could still occur in the datastream are:

- Numeric Space, X'E1'
- Required hyphen, X'60'
- Required Space, X'41'
- Syllable hyphen, X'CA'

3. In the DBTTRN1 program, the actual transformation of 1403 print data to DCA Level 2 is done. The DCA Level 2 controls used should provide document integrity for 1403 print data. After finding the print lines in the blocked record the IRS codes are removed. 1403 print controls are converted as follows:

Meaning	Input	Output L2DCA stream
1403 Printer Controls		
Space 1 line after printing	X'09'-prtline	X'0D'-prtline-X'15'
Space 2 lines after printing	X'11'-prtline	X'0D'-prtline-X'1515'
Space 3 lines after printing	X'19'-prtline	X'0D'-prtline-X'151515'
Skip to channel 1 after prt	X'89'-prtline	X'0D'-prtline-X'0C'
Skip to channel 2 after prt	X'91'-prtline	X'0D'-prtline-X'1515'
.....	....	X'0D'-prtline-X'1515'
Skip to channel 12 after prt	X'E1'-prtline	X'0D'-prtline-X'1515'
Space 1 line immediate	X'0B'-prtline	X'15'-prtline
Space 2 lines immediate	X'13'-prtline	X'1515'-prtline
Space 3 lines immediate	X'1B'-prtline	X'151515'-prtline
Skip to channel 0 immediate	X'83'-prtline	X'0C'-prtline
Skip to channel 1 immediate	X'8B'-prtline	X'0C'-prtline
Skip to channel 2 immediate	X'93'-prtline	X'1515'-prtline
.....	....	X'1515'-prtline
Skip to channel 12 immediate	X'E3'-prtline	X'1515'-prtline
Write without spacing	X'01'-prtline	X'0D'-prtline
No-op	X'03'-prtline	delete prtline
Anything else	-prtline	X'15'-prtline
ANSI Print Controls		
Space 1 line before printing	X'40'-prtline	X'15'-prtline
Space 2 lines before printing	0 -prtline	X'1515'-prtline
Space 3 lines before printing	- -prtline	X'151515'-prtline
No space before printing	+ -prtline	X'0D'-prtline
Start new page	1 -prtline	X'0C'-prtline

The X'0D' controls at the beginning of 1403 print controls that are effective after printing ensure that if a previous print line had an 'immediate' print control, these print lines still will start printing in position 1 of the print line.

At the end of the document, the last print line will have no NL (X'15') appended to it, since this could overflow the presentation space. Instead the document will end with ZICR (X'0D') and FF (X'0C'). The reason for this is that Displaywriter and DOSF do not always start a new document with a form feed as SCRIPT output normally does; they end a document with ZICR and FF instead. Documents without ZICR and FF at the end will give an error message on Displaywriter when printed.

Additionally, some initial settings for formatting are necessary with multi-byte controls. The following settings are provided by DBTTRN1:

- SEA, set exception action, X'2BD2nn85' where nn is a count field. We have set the exception class and action bytes to X'0000' which means: for all exception classes - still present the data but indicate loss of fidelity and possible alternatives. If print fidelity is required this should be set to: X'000101020202'. No data will be presented if loss of text data or loss of appearance would occur.

- SHM, set horizontal margins, X'2BD2nn11'. We have set the left margin to 0.0 inch and the right margin to 8.5 inches. The right margin setting is ignored because set justify mode is not used. The left margin is set to 0 to ensure print fidelity with the original document. The operand field is then: X'00012FD0'.

- SVM, set vertical margins, X'2BD2nn49'. We have set top margin to 0.5 inches as SCRIPT assumes that a 'skip to channel 1' will actually be on the 4th print line of a new page. The bottom margin is set to 11 inches. Bottom margin is ignored in DCA Level 2. The operand field is then: X'02D03DE0'.

- SPPS, set presentation page size, X'2BD2nn40'. We have set the width to 8.5 inches and the page depth to 11 inches. The operand field is then: X'2FD03DE0'.

- SCG, set CGCSGID[4], X'2BD10601' The CGCSGID is set to X'00D70108' (215-264 when expressed in decimal). This corresponds to the GCID reserved in DISOSS for the TN-chain of the 1403 printer.

3.2.2.2 Character Translations

Terminology

The L2DCA SCG control introduces the term CGCSGID (Coded Graphic Character Set Global ID). The CGCSGID is a definition of the relationship between the hexadecimal codepoints in the datastream and the graphic characters presented on a display or printer. CGCSGID is made up of two components:

GCSGID Graphic Character Set Global ID. A two-byte field identifying a predefined and documented set of graphic characters. This set could be, for example, the characters available on a particular keyboard on on a particular printwheel.

CPGID Code-Page Global ID. A two-byte field identifying a codepage; a codepage defines the graphic character to be displayed for each of the 256 possible hexadecimal codepoints.

There exist many more than 256 graphic characters which may need to be displayed, so the CPGID provides a means of identifying which set is to be used in a particular datastream, and which hexadecimal codepoints are to represent them.

[4] The term CGCSGID, used in the L2DCA architecture, is equivalent to the term GCID used in DISOSS publications. Their meaning is described in "Terminology."

Similarly, most displays and printers cannot support as many as 256 graphic characters at any given moment, so the GCSGID provides a means of selecting a subset from those characters available on the codepage.

The terms used here are those used in the L2DCA architecture; other terms are used in other publications to refer to the same definitions. See Figure 11.

Full Name	Possible Abbreviations		
	L2DCA	L3DCA	DISOSS
Coded Graphic Character Set Global ID.	CGCSGID	GCID	GCID
Graphic Character Set Global ID.	GCSGID	CGCS ID	GGID
Code Page Global ID.	CPGID	Code Page ID	GPID

Figure 11. Graphic Character Set Definitions: different terms may be used to refer to the same entity.

For brevity, this book uses the terms GCID, GGID and GPID.

How DISOSS Chooses Character Translations

The procedure used by DISOSS to choose a translate table for an output document is as follows:

The GCID list provided by the output device at DIA Sign_On is searched for the output document GCID. If the search is successful, no translation occurs.

Otherwise, the translate table index is searched for a translate table suitable for the document GPID, and an output device GCID. If the search is successful, then that table is used.

Otherwise, no translation occurs.

Required Character Translations

There is a GCID known to DISOSS which represents the characters on the 1403 TN print train; this is X'00D7 0108' (00215-00264). Clearly, if we describe our input document with this GCID, then its content is accurately identified, and all components of the network have the means of knowing what our text really is. The disadvantage is that few components of today's DISOSS networks were designed to handle this GCID: thus Displaywriter, Scanmaster, DISOSS/8100 and DISOSS/PS will all either reject a document using this GCID, or will print it incorrectly. Only the DISOSS Host Print function can handle it as intended.

This is not an unexpected situation in a DISOSS system; there are many cases in which a document is to be delivered to a subsystem which does not support the document's GCID, and DISOSS provides a set of translate tables which it uses to translate from the input GCID to a GCID acceptable to the receiver. DISOSS also provides a way for an installation to add its own translate tables to the standard ones. So our solution to the present problem is to provide a translate table to map the 1403 TN characters on to a GCID that is understood by all of the likely receiving subsystems. We could alternatively have performed a translation in DBTTRN1, before passing the document into DISOSS, but rejected this approach for two reasons:

1. If DISOSS will perform the translation for us, there seems no point in duplicating the function.

2. The translation cannot completely retain the appearance of the original document (certain box junction characters are lost, for example), and so it is preferable to translate only when necessary: using the DISOSS translate function ensures that translation occurs only when the document is about to be output to a device that needs it. While stored in the library, or when delivered to a recipient (such as another CMS/PROFS user) who can handle the original GCID, the document does not undergo any translation and retains its original appearance.

The output GCID we selected is X'0151 0100' (00337-00256), which is the Multi-Lingual Codepage and is supported by all DISOSS subsystems. The translate table we have set up from GPID X'108' (264) to GCID X'01510100' (337-256) tries to preserve as much of the meaning of the printable graphics as possible. When no similar graphic could be found on code page X'100', a substitute was chosen. See "DBTTRT01 Translate Table" on page 99.

3.2.2.3 Overview of Transformations and Translations

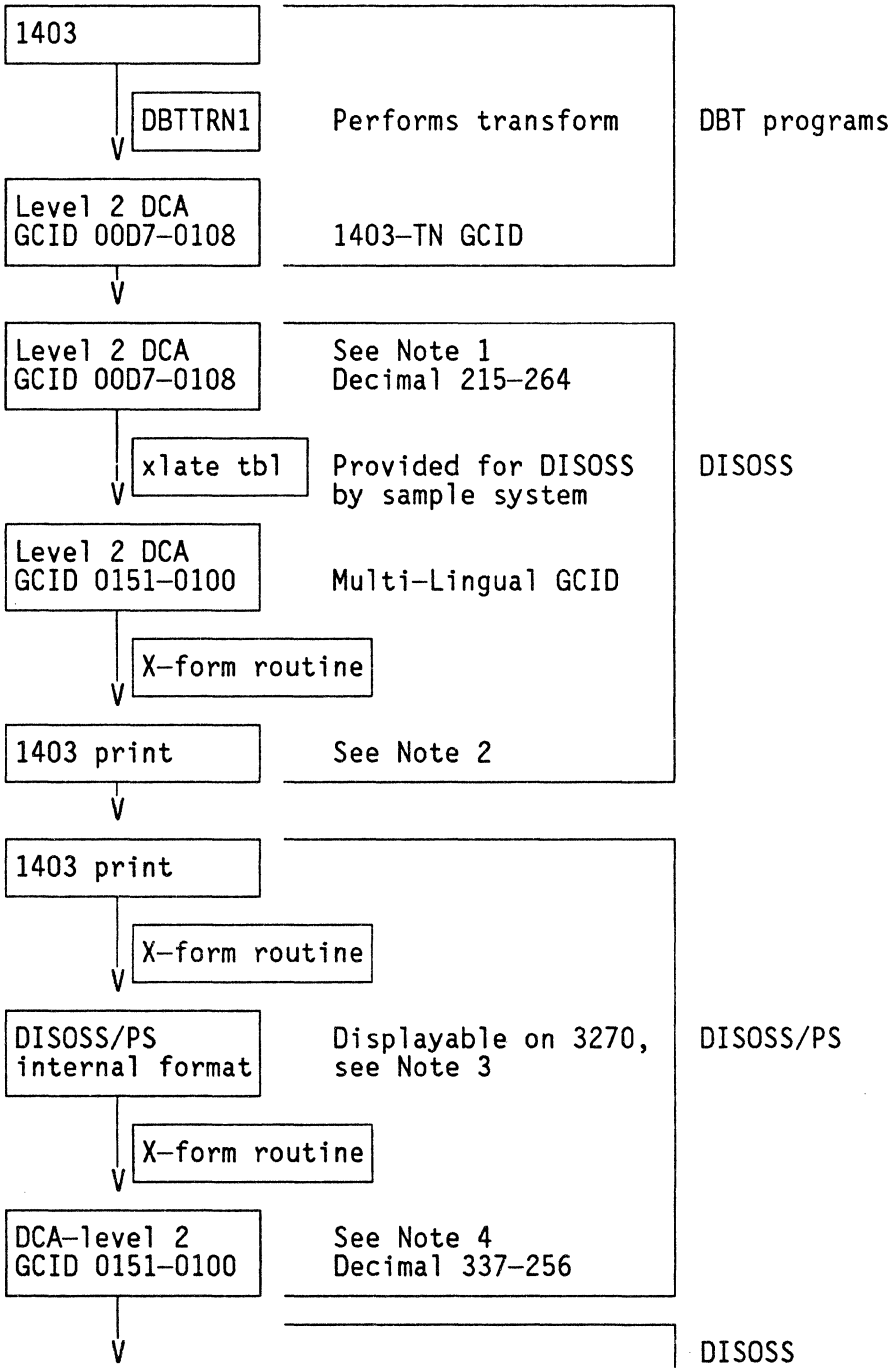


Figure 12. Transformations and Translations: as a document passes through the system, it may be transformed and translated several times.

Notes

1. At this moment the document still conserves its print fidelity. If it were filed now, it would be stored in the library with print fidelity maintained.

2. As DISOSS/PS at signon time declares it can handle '1403' type documents, DISOSS schedules the appropriate transform routine. The document will thus be delivered to DISOSS/PS in a form equivalent to the original 1403 print output of Box X, apart from the character translation we provided.

3. DISOSS/PS will do a transform to an internal format to be able to display documents on a 3270 screen. This means it will, amongst other things, delete overprinted lines.

4. When DISOSS/PS again gives the document to DISOSS (for a file, distribute etc.), it first transforms the document to a L2DCA format.

The last two transforms do not preserve print fidelity. The following changes will occur compared with the L2DCA document created by our program DBTTRN1:

- The multibyte controls at the beginning of the DCA stream will be replaced by the controls provided by DISOSS/PS.
- NL controls (X'15') are replaced by RNL controls (X'06').
- FF controls (X'0C') are replaced by RFF controls (X'3A').
- ZICR controls (X'0D') are deleted.
- Every overprinted line is deleted.
- The document ends with RNL,FF.

The main consequences of this to a DCF-generated document are:

- Box corner characters become full-stops.

- Overstruck lines are lost. This means:

  - DCF titles are no longer bold.
  - All underscoring is lost.
  - Box intersection characters, which are made up of one character overprinted on another, are lost, and the box is incorrectly formed.

Simple memos, or output from programs other than DCF, may not be seriously affected by these losses, but in order to allow complex DCF documents to be handled by our system, we had to provide an additional function. It is possible for the CMS/PROFS user to request that the document be filed on behalf of a DISOSS/PS user (usually himself), instead of being distributed to that DISOSS/PS user. In this way, the document is not sent to the DISOSS/PS user's Mail Log, but remains intact as a L2DCA document in the DISOSS library. The DISOSS/PS user can then search for it, add search terms and access codes if necessary, distribute it to other DISOSS users, or delete it.

3.2.3 BATCH-CICS INTERFACE

A VSAM file, which we call DBTVSQ0, is chosen as the vehicle to move documents from the RJE system into the CICS environment. Control of access to this VSAM file is exercised through Open/Close processing and VSAM Shareoptions set to 1 (which allows only one concurrent user). Both the batch program DBTBAT1 and the CICS transaction DBTMOV1 will try to open this file for as short a period of time as possible. If DBTBAT1 does not succeed on the first attempt, it will retry the open until it is successful. In the same circumstances, DBTMOV1 will end and the next initiation of DBTMOV1 will pick up any accumulated documents. Most of the time this dataset will be closed to CICS. DBTMOV1 copies the contents of DBTVSQ0 to an identical but non-shared file, DBTVSQ1.

DBTVSQ0 has variable length records with a maximum record size of 6000 bytes (which fits well on most type of DASD). A typical 2-3 page document will then only take up one record on this dataset.

DBTVSQ0 and DBTVSQ1 are key sequenced to simplify possible future modifications where records may not be entered sequentially, or where concurrent applications may be writing to the same dataset. The common DBTVSQ0/DBTVSQ1 key fields are as follows:

FIELD NAME	SIZE IN BYTES	DESCRIPTION
DATE	5	Date
TIME	9	Time
INTTYPE	1	Interface Type identifier
OSN	8	Office System Name
USER	8	User name
INTYPE	2	Input document type
OUTYPE	2	Output document type
SEQNO	2	Sequence Number
CHFLAGF	1	Chain flag first
CHFLAGL	1	Chain flag last
TOTAL	39	

DATE This is the date in YMMDD form. Y is the least significant year digit.

TIME This is the time in hhmmssttt form. 'ttt' is the milliseconds.

INTTYPE This identifies the interface into our system. Only one is currently defined (the DBTVSQ0 shared dataset), but others might be required in the future. For example, some subsystems could have an SNA session with CICS, across which documents could be transferred; the receiving CICS transaction could insert the document in the CICS dataset DBTVSQ1, but would indicate in the document header INTTYPE field that the document arrived via a different interface.

The combined DATE/TIME/INTTYPE field is used as a unique request identifier, and this should ensure that each document is stored in a series of records in ascending key sequence, and that duplicate keys cannot occur. This depends on the assumption that the probability of more than one batch job using the PL/I TIME pseudovariable in the same millisecond, is negligible. If this assumption is not considered satisfactory, the problem could be avoided by ensuring that the batch jobs do not execute concurrently, or that they serialise on some common resource before taking the timestamp.

OSN This is the name of the distribution node of the recipient.

USER This is the name of the recipient.

INTYPE This identifies the datastream type of the document text. Currently valid values are:

- X'000C' --- 1403 print lines
- X'0002' --- DCA Level 2

OUTYPE This identifies the desired datastream type. Currently valid values are:

- X'0002' --- DCA Level 2

If INTYPE is not equel to OUTYPE, then this is an indication that the document requires transformation.

SEQNO This is a binary number one less than the number of the record. The existence of this field ensures the impossibility of duplicate keys from the same request, and also causes the request records to be arranged in order by VSAM.

CHFLAGF This is '1' if the record is the first of a document, and '0' if the record is not the first. All other values are invalid. It is used to identify the beginning of a document.

This field is strictly unnecessary, since its value can always be deduced from SEQNO. It is included however, to compartmentalise the functions of the fields, and to make the code which manipulates the fields more easy to follow.

CHFLAGL This is '1' if the record is the last of a document, and '0' if the record is not the last. All other values are invalid. It is used to identify the end of a document.

This field might be used by a future version of DBTMOV1, if the design were changed to allow concurrent CICS/Batch access to the shared dataset. In that case, DBTMOV1 would not wish to start reading a document until it knew that DBTBAT1 had finished writing it.

The key contains the minimum distribution information. It could be used in a future design to allow partial distribution of a document in case of errors.

Additional distribution information is present in an 80-byte header field. This field is the first card in the batch SYSIN stream. It is included in the first and last VSAM record starting at byte number 42. There is currently still plenty of space for more additions. The fields are as follows:

FIELD NAME	SIZE IN BYTES	DESCRIPTION
RECTYP	1	Batch SYSIN record type
PROFLAG	1	Profile format indicator
PAGEL	3	Pagelength
PAGEW	3	Pagewidth
DISNAM	8	Not used
EYECAT	6	"HEADER" Eyecatcher
DOCNAM	15	Document name
DISFIL	1	DIA Command - Req_Dist. or File
RESER	42	Not used
TOTAL	80	

Below are some explanations of the header fields:

RECTYP Batch input record format identifier. Would be used by DBTBAT1 if it supported more than 1 input record format.

PROFLAG Profile format identifier. In our case, there is only one format, contained on one card. Would be used by all programs in the request processing flow if more than one profile format was supported. Additional profile formats would be required if and only if the total size of possible profile parameters exceeded 80 bytes.

PAGEL Page length. Could be used by DBTTRNn if the input pagelength were not ignored, or if the output pagelength were not preset. The precise meaning of this field would depend on the transform.

PAGEW Page width. Could be used by DBTTRNn if the input pagewidth were not ignored, or if the output pagewidth were not preset. The precise meaning of this field would depend on the transform.

DISNAM Not used.

DOCNAM Document name. Up to 15 characters.

DISFIL Allows the user to select the DIA command to be built; valid values are 'D', for a Request_Distribution command, and 'F' for a File command.

RESER Not used.

3.2.4 COMPONENTS OF THE BOX X TO DISOSS FACILITY.

3.2.4.1 DBTBAT1

Batch PL/I program DBTBAT1 is executed by the procedure invoked by the RJE batch job. Its function is to write a document input request to DBSVSQ0.

Input: JCL and instream data containing:

- Header with user, profile, and processing information.
- Chopped up printlines.

Output: 6000 byte DBTVSQ0 records containing:

- Key with user and processing information.
- Profile information on first and last record.
- IRS separated printlines

3.2.4.2 DBTMOV1

CICS PL/I program DBTMOV1, the only program of the DBTM transaction, moves records from the CICS-Batch shared dataset DBTVSQ0, to CICS dataset DBTVSQ1. It calls subroutine DBTOPN1 to open DBTVSQ0 to CICS, and calls DBTCLS1 to close DBTVSQ0 from CICS.

No transformations are done.

Copied records are deleted form DBTVSQ0.

A DBTS transaction is initiated for each request, with the key of the first record as start data to help the transform selection program locate the document.

The program issues a delayed start of its own transaction to cause its periodic re-initiation.

3.2.4.3 DBTOPN1

CICS assembler program DBTOPN1 issues a DFHOC OPEN macro for dataset DBTVSQ0 on behalf of DBTMOV1.

3.2.4.4 DBTCLS1

CICS assembler program DBTCLS1 issues a DFHOC CLOSE macro for dataset DBTVSQ0 on behalf of DBTMOV1.

3.2.4.5 DBTMST1

CICS PL/I program DBTMST1, the first program of the DBTS transaction, retrieves the first-in-chain key passed from DBTMOV1, and uses it to obtain the whole FIC record. The fields of the record are analysed to select an appropriate DBTTRNn transform routine, which is started with the FIC key as start data, to help locate the document in DBTVSQ1.

3.2.4.6 DBTTRN1

CICS PL/I program DBTTRN1 is the only transform program in the sample system. The input is a set of 6000 byte KSDS VSAM records from DBTVSQ1, with:

- 39 byte key with user and processing data.
- Profile data on FIC and LIC record.
- IRS separated printlines.

The output is a set of 4088 byte KSDS VSAM records to DBTVSQ1, with:

- 39 byte key with user and processing data.
- Profile data on FIC and LIC record.
- L2DCA datastream.

The first input record is located by the start data received from DBTMST1.

The document is converted to a DCA Level 2 datastream and is written to DBTVSQ1 in units of a convenient size for the API queue.

The input records are deleted from DBTVSQ1.

Program DBTSND1 is started with the new FIC key as start data.

3.2.4.7 DBTSND1

CICS PL/I program DBTSND1 sends a DIA Request_Distribution or File command to the DISOSS API.

The document is located on DBTVSQ1 using the FIC key retrieved from DBTTRN1.

A profile parameter block is constructed from the header.

The DIU text segments are transmitted to the API. Each segment corresponds to one input record.

An API 'Last' command is transmitted to initiate DISOSS processing.

The input records are not deleted from DBTVSQ1. This is a function of the response transaction. This aids problem determination by preventing the deletion of the transformed request in the event of an error.

3.2.4.8 DBTRSP1

CICS PL/I program DBTRSP1 is the only program of the response transaction DBTR. If the DISOSS response is normal, the request is deleted from DBTVSQ1. If the response is not normal, an API PURGE is issued.

3.2.4.9 DBTSON1

CICS PL/I program DBTSON1, the only program of the DBTN transaction, does a DIA 'Sign_On' to DISOSS. It is executed twice in the 'Sign_On' process.

In the first execution it sends a 'Sign_On' to DISOSS, naming itself as the response transaction.

In the second execution it starts the DBTM cycle if the DISOSS response is normal, and issues an API PURGE if the DISOSS response is not normal.

Instead of being invoked via DBTN, this program can be included in the CICS PLT, and can thus be executed at CICS start-up.

3.2.4.10 DBTCLN1

CICS PL/I program DBTCLN1, the only program of the DBTC transaction, reinitialises the datasets DBTVSQ0 and DBTVSQ1.

The program is not strictly required, but it is useful, especially in a development enviroment, because it enables these datasets to be reinitialised conveniently while CICS is up.

3.2.4.11 DBTSOF1

CICS PL/I program DBTSOF1, the only program in the DBTF transaction, does a DIA 'Sign_Off' from DISOSS. It is executed twice in the 'Sign_off' process.

In the first execution it sends a 'Sign_Off' to DISOSS, naming itself as the response transaction. In the second execution, it issues an API PURGE if the DISOSS response is not normal.

This program is not used in the sample implementation.

4.0 SYSTEM DEFINITIONS FOR THE SAMPLE IMPLEMENTATION

This section deals with the system definitions used in the course of the project to verify the design. Some knowledge of CICS and DISOSS table generation is assumed.

4.1 CICS TABLES

These tables are needed for the Box X to DISOSS (inbound) function, and are not used for the DISOSS to Box X (outbound) function.

4.1.1 FILE CONTROL TABLE

Below are the entries made for the VSAM datasets DBTVSQ0 and DBTVSQ1.

```
DFHFCT TYPE=DATASET,
      DATASET=DBTVSQ0,
      ACCMETH=(VSAM,KSDS),
      SERVREQ=(UPDATE,NEWREC,BROWSE,DELETE),
      FILSTAT=(ENABLED,CLOSED),
      RECFORM=(VARIABLE,UNBLOCKED),
      BUFND=3,BUFNI=2,
      STRNO=2,
      MODE=VSAM

DFHFCT TYPE=DATASET,
      DATASET=DBTVSQ1,
      ACCMETH=(VSAM,KSDS),
      SERVREQ=(UPDATE,NEWREC,BROWSE,DELETE),
      FILSTAT=(ENABLED,OPENED),
      RECFORM=(VARIABLE,UNBLOCKED),
      BUFND=12,BUFNI=10,
      STRNO=10,
      MODE=VSAM
```

The BUFND, BUFNI, and STRNO parameters determine the DBTVSQ1 buffer allocation. Their values should be considered carefully.

4.1.2 PROGRAM CONTROL TABLE

Below are the entries made for the CICS transactions.

```
DCFPCT TYPE=ENTRY,TRANSID=DBTN,PROGRAM=DBTSON     SIGNON
DCFPCT TYPE=ENTRY,TRANSID=DBTM,PROGRAM=DBTMOV1    MOVE
DCFPCT TYPE=ENTRY,TRANSID=DBTS,PROGRAM=DBTMST1    SEND
DCFPCT TYPE=ENTRY,TRANSID=DBTR,PROGRAM=DBTRSP1    RESPONSE
DCFPCT TYPE=ENTRY,TRANSID=DBTF,PROGRAM=DBTSOF     SIGNOFF
DCFPCT TYPE=ENTRY,TRANSID=DBTC,PROGRAM=DBTCLN     CLEANUP
```

4.1.3 PROGRAM LIST TABLE

A startup PLT is a list of programs to be automatically initiated during CICS startup. The DIA Sign_On program DBTSON1 is executed here, to ensure that a DIA session exists with DISOSS before any DBTS transactions attempt to distribute documents.

If the CICS system already has a startup PLT, the entry for DBTSON1 can be added to it; otherwise, a new table can be created as follows:

- A PLT table, similar to the one below, must be assembled.
- The table must be defined in the PPT, as described in the section about the PPT in this chapter.
- The table suffix must be specified in the SIT, as described in the section about the SIT in this chapter.

Below is a sample PLT used for the initiation of DBTSON1.

```
         DFHPLT TYPE=INITIAL,SUFFIX=ST
         DFHPLT TYPE=ENTRY,PROGRAM=DBTSON1
         DFHPLT TYPE=FINAL
         END
```

4.1.4 PROGRAM PROCESSING TABLE

Below are the entries for the CICS programs in the PPT.

```
DFHPPT TYPE=ENTRY,PROGRAM=DBTSON1,PGMLANG=PL1    SIGNON
DFHPPT TYPE=ENTRY,PROGRAM=DBTMOV1,PGMLANG=PL1    MOVE FROM BATCH
DFHPPT TYPE=ENTRY,PROGRAM=DBTMST1,PGMLANG=PL1    SELECT TRANSFORM
DFHPPT TYPE=ENTRY,PROGRAM=DBTTRN1,PGMLANG=PL1    1403 --> L2DCA
DFHPPT TYPE=ENTRY,PROGRAM=DBTSND1,PGMLANG=PL1    SEND TO API
DFHPPT TYPE=ENTRY,PROGRAM=DBTRSP1,PGMLANG=PL1    API RESPONSE
DFHPPT TYPE=ENTRY,PROGRAM=DBTOPN1                OPEN  DBTVSQ0
DFHPPT TYPE=ENTRY,PROGRAM=DBTCLS1                CLOSE DBTVSQ0
DFHPPT TYPE=ENTRY,PROGRAM=DBTSOF1,PGMLANG=PL1    SIGNOFF
DFHPPT TYPE=ENTRY,PROGRAM=DBTCLN1,PGMLANG=PL1    CLEANUP
```

Below is the entry for the startup PLT. This only has to be done if a new startup PLT has to be created.

```
DFHPPT TYPE=ENTRY,PROGRAM=DFHPLTST               STARTUP PLT
```

Below is an entry for a print/translate table

```
DFHPPT TYPE=ENTRY,PROGRAM=DBTTRT01,PGMLANG=ASSEMBLER,
            PGMSTAT=ENABLED,RELOAD=NO,RES=NO
```

4.1.5 SYSTEM INITIALISATION TABLE

Below is the DFHSIT TYPE=CSECT macro option which indicates the startup program list table suffix.

```
                PLTPI=ST,
```

If desired, reassembly of the SIT table can be avoided, by specifying the PLTPI option within a PARM parameter in the CICS startup JCL.

4.2 VSAM DATASET DEFINITIONS

The VSAM datasets DBTVSQ0 and DBTVSQ1 were created using a job with the following three steps:

1. Delete the datasets in case they already exist

2. Allocate the datasets.

3. Initialise the datasets using the IDCAMS REPRO utility to write a dummy record. The first 39 bytes of the dummy records, which form the key, were all set to X'FF'. This ensures that the dummy record is always pushed to the end of the file when data is added.

Below is the SYSIN data for the IDCAMS allocate step.

```
DEFINE CLUSTER -
         (NAME(DISOSS30.DBTVSQ0) -
         VOL(WTL372) -
         CYLINDERS (2 1) -
         KEYS(39 0) -
         RECSZ(5000 6000) -
         SHAREOPTIONS (1) -
         UNIQUE) -
       CATALOG(VWTL372) -
       DATA -
         (NAME(DISOSS30.DBTVSQ0.DATA)) -
       INDEX -
         (NAME(DISOSS30.DBTVSQ0.INDEX))
DEFINE CLUSTER -
         (NAME(DISOSS30.DBTVSQ1) -
         VOL(WTL372) -
         CYLINDERS (2 1) -
         KEYS(39 0) -
         RECSZ(5000 6000) -
         SHAREOPTIONS (1) -
         UNIQUE) -
       CATALOG(VWTL372) -
       DATA -
         (NAME(DISOSS30.DBTVSQ1.DATA)) -
       INDEX -
         (NAME(DISOSS30.DBTVSQ1.INDEX))
```

4.3 DISOSS TABLE DEFINITIONS (BOX X TO DISOSS)

4.3.1 TRANSLATE TABLES

Most devices which are to receive L2DCA documents containing the 1403 TN character set will require a suitable translate/print fidelity table; we provided a table called DBTTRT01, which translates from 1403 to the Multi-Lingual Codepage. Installing this table in DISOSS involves two steps:

1. Assemble and link the translate table itself. The best way to create a new table is to modify a copy of a previously existing job; DISOSS provides samples on the installation tape. The job we used is shown in "DBTTRT01 Translate Table" on page 99.

2. Make the new table known to DISOSS by adding an entry into the index table DSVS5800. Again, a sample job is provided by DISOSS; our new entry was:

```
DSVXIDX TYPE=ENTRY,INGPID=00264,OUTGCID=00337-00256,
                TBLID=DBTTRT01
```

4.3.2 HOST USER PROFILE

Below is an example of the HUP definition for the mailbox API user represented by our programs. The FORUSER parameter authorises DIST01 to file documents on behalf of DISOSS/PS user PSUSER01.

```
ADD USERTYPE=API,
    EXTERNAL='Mailbox API User 1      ',
    REQPWD='1',DDN='DSVHOST',SA='DIST01',
    FORUSER=(DSVHOST,PSUSER01)
```

4.4 DISOSS TABLE DEFINITIONS (DISOSS TO BOX X)

These HUP and PDT entries are used by the DISOSS to Box X (outbound) function. They are not used by the Box X to DISOSS (inbound) function.

4.4.1 HOST USER PROFILE

If the jobcard parameters in the JOBJCL option of the DISOSS Host Definition job have not been set or are insufficient, then extra Host Print jobcard parameters may be specified in the accounting information field of the DISOSS user Host User Profile. Below is an example:

```
ACCOUNT='(P-032007),MSGCLASS=A,MSGLEVEL=(1,1),CLASS=A',
```

4.4.2 PRINTER DESCRIPTION TABLE

Box X destinations must be defined as printers in the PDT. The PDT specifies the page width, page length, and a print/format procedure. Below is a sample entry:

```
  DBTPRT  DSVPDT TYPE=ENTRY,PRTTYPE=PRINTER,                             X
                 LINEWD=132,PAGEDP=66,                                   X
                 JOB=DBTPRT
```

If Box X has RJE output support for multiple destinations (as, for example the VM support for a virtual reader for each user), then multiple PROCs will be required: one for each Box X destination. They could all be invoked via this single entry in the PDT, however; the DISOSS user would specify printer DBTPRTnn (where nn is two numeric digits) and DISOSS would invoke a JCL PROC called DBTPRTnn. Only the single entry for DBTPRT is needed in the PDT.

4.5 PRINT/FORMAT PROCEDURE

Below is an example of a simple print/format procedure. The procedure could be customised to produce output specific to Box X. As an example, the document could be converted to a PROFS note, by using the IDCAMS REPRO utility to wrap a header and footer around the formatter output.

```
//DBTPRT99 PROC DSVCOPY=1,
//             DOCNODE=RALYDPD3,DOCUSER=HAY,       DOCUMENT DESTINATION
//             MSGNODE=RALVSMV3,MSGUSER=RMT99      JCL, MESSAGES ETC.
//*
//FORMAT  EXEC PGM=DSVOL500
//STEPLIB   DD DSN=DISOSS30.DSVLOAD,DISP=SHR
//*
//DOC   OUTPUT COPIES=&DSVCOPY,DEST=&DOCNODE..&DOCUSER
//MSG   OUTPUT DEFAULT=YES,DEST=&MSGNODE..&MSGUSER
//*
//DSVPRINT  DD SYSOUT=A,DCB=(RECFM=FA,LRECL=133),OUTPUT=(*.DOC)
//DSVMSG    DD SYSOUT=A,DCB=(RECFM=FA,LRECL=133)
//DSVDUMP   DD SYSOUT=A
```

This procedure makes use of the OUTPUT JCL statement available with MVS/SP Version 1 Release 3.3. This allows NJE output routing information to be included in a PROC, which was not possible with the earlier JES2 /*ROUTE statement.

4.6 IMPROVEMENTS AND ALTERNATIVE OPTIONS

A limit to the load on CICS can be set by specifying the maximum number of concurrent active Mailbox tasks. This can be done by assigning a transaction class to Mailbox transactions in the PCT, using the TCLASS parameter, for example TCLASS=7, and by specifying a class activation ceiling in the SIT using the CMAT option, for example CMAT=(,,,,,,9,,,).

It was noticed during program development that when the number of DBTS transactions exceeded the available DBTVSQ0/1 buffer allocation, they always hung as if in a deadlock, until enough had been force-purged to allow the others to continue execution. This may have been the result of bad system tuning, but since it has occurred in one system, it could occur in others. For this reason a maximum active task limit is a worthwhile precaution against overloading the available buffers.

5.0 COMMUNICATING BETWEEN PROFS AND DISOSS

5.1 OVERVIEW

PROFS (Professional Office System) runs under VM, and uses CMS facilities to create, file and retrieve notes and documents.

We have attempted to provide a "bridge" between PROFS and DISOSS that would allow the PROFS user access to all the functions of DISOSS. The examples given here are for guidance only, an account of how the problem was tackled at the Raleigh International Systems Centre.

The steps involved in this part of the exercise are:

1. Invoking DISOSS/PS from PROFS

   Some small EXECs are given, showing the user how to log on to DISOSS/PS from the PROFS main menu, send files from the A-disk to DISOSS and to use RDRLIST to receive and edit documents sent from DISOSS.

2. Sending PROFS documents to DISOSS.

   An explanation is given on the difference between PROFS "final" documents and "draft" documents. A method is given explaining how to copy a draft document to a final document before sending it to DISOSS.

3. Sending PROFS notes to DISOSS.

   PROFS notes are unformatted and contain some characters not needed by DISOSS. A sample EXEC is shown, demonstrating how to edit these files, getting rid of the unwanted lines and hex characters, and then submitting the edited note to the DBTSEND EXEC, which passes it on to DISOSS.

4. Loading DISOSS documents to the CMS A-disk and moving them to PROFS.

   This part explains how to load a document sent to the PROFS/CMS user's virtual reader from DISOSS.

5.2 ACCESSING DISOSS/PS FROM PROFS

To access DISOSS/PS from PROFS, we made a simple change to the OFS $SYSPROF file on the SYSADMIN 399 disk.

For example:

```
SET TITLE                          PROFESSIONAL OFFICE SYSTEM
SET MENU 1
SET PF1 'APPOINTM' Process schedules
SET PF2 'OPENMAIL' Open the mail
SET PF3 'SEARCH' Search for documents
SET PF4 'OFSNOTE' Process notes and messages
SET PF5 'MEMO' Prepare documents
SET PF6 'SET FILEDOCU' Process documents from other sources
SET PF7 'MAILLOG' Process the mail log
SET PF8 'MAILMAN STATUS' Check the outgoing mail
SET PF10 'DISOSS' DISOSS tasks
SET PF11 'SET MENU 2' Look at main menu number 2
SET MENU 2
SET PF11 'SET MENU 3' Look at main menu number 3
SET MENU 3
SET PF11 'SET MENU 1' Look at main menu number 1
SET MENU 1
```

The DISOSS option accessed by **PF10** in PROFS main menu 1 invokes a simple EXEC (called DBTMENU) that presents this menu:

```
Send a Document to DISOSS Users         -  1
Logon to DISOSS/PS                      -  2
Read in and send a Note to DISOSS Users -  3
Receive a Document from DISOSS          -  4
```

- Option 1, "Send a Document to DISOSS Users", builds a batch job and sends it via RJE to MVS.

- Option 2, "Logon to DISOSS/PS", allows the user to sign on to DISOSS/PS via the VM PASSTHRU program.

- Option 3, "Read in and send a Note to DISOSS Users", invokes RDRLIST from where the user enters the "DBTNOTE" command, as explained in "The DBTNOTE EXEC" on page 59.

- Option 4, "Receive a Document from DISOSS", invokes RDRLIST from where the user enters the "DBTRECV" command to read a DISOSS document on to his A-disk.

The DBTMENU EXEC is shown in "Sample DBTMENU EXEC" on page 155.

The DBTSEND EXEC (Option 1), after asking the user for the filename, filetype and filemode, then takes a 1403 print file (usually a document extracted from the PROFS library), encapsulates it in an MVS job and submits it to the MVS system for input to DISOSS. An example of this EXEC is shown in "Sample DBTSEND EXEC" on page 157.

The DBTLOGON EXEC (Option 2) invokes VM PASSTHRU and gives the user direct access to CICS so that he can log on to DISOSS/PS. An example of this EXEC is shown in "Sample DBTLOGON EXEC" on page 161.

RDRLIST, invoked by Options 3 and 4, is a standard VM/SP2 facility which displays a list of the files in the virtual reader, and allows the user to enter commands alongside the name of a file on the list. In this case, the user enters "DBTNOTE" or "DBTRECV" and presses PF10.

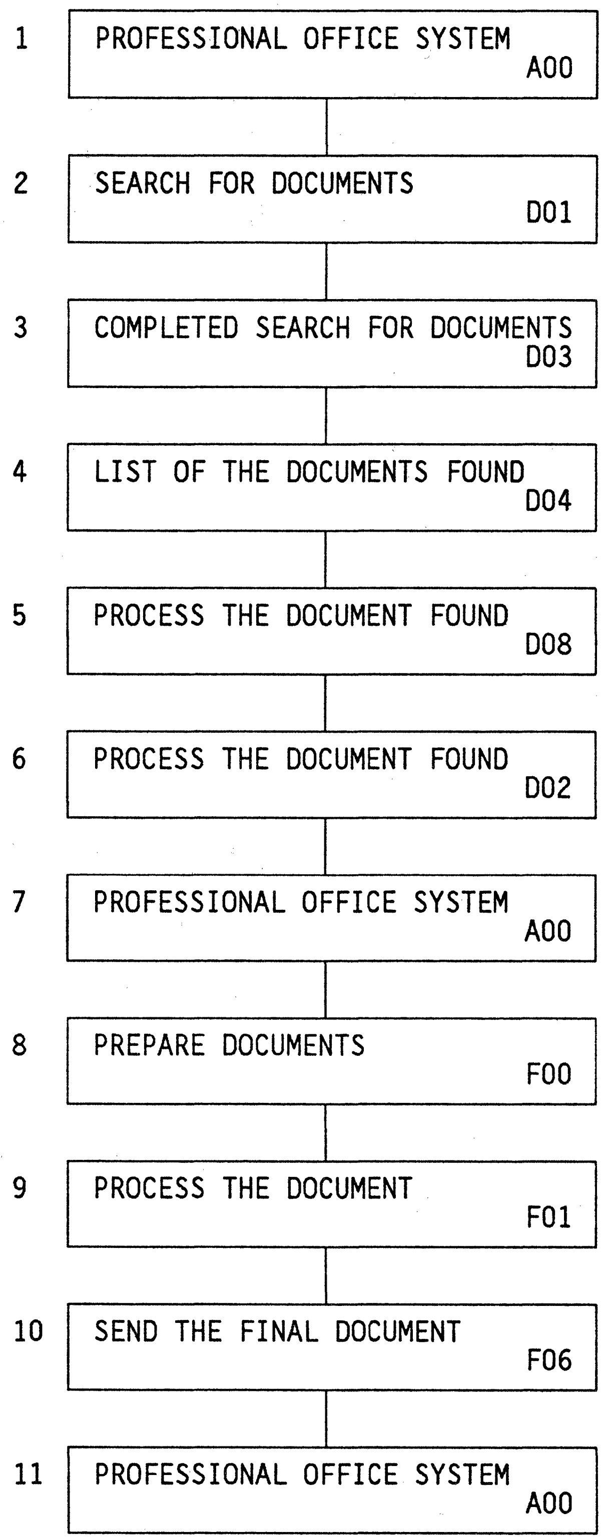
An example of DBTRECV is shown in "Sample DBTRECV EXEC" on page 164. An example of DBTNOTE is shown in "Sample DBTNOTE EXEC" on page 163, and an explanation of its function is given in "The DBTNOTE EXEC" on page 59.

5.3 SENDING A PROFS DOCUMENT TO DISOSS

The diagram on the following page illustrates the steps involved in searching for a PROFS document and copying the document on to your A-disk, ready to be sent via DBTSEND to DISOSS.

Our method assumes that the document being sent to DISOSS is in 1403 final form; that is, any SCRIPT control words and GML tags have been resolved. Therefore, only a PROFS "final" document is sent, not a "draft" document.

If you need to send a PROFS **draft** document to DISOSS, follow the steps outlined in "Changing a DCF file to 1403 Format" on page 60; this will take a copy of your draft document, convert it to final form, then send it to DISOSS.



1 PROFESSIONAL OFFICE SYSTEM A00
Search for documents
Press PF3

2 SEARCH FOR DOCUMENTS D01
Enter search terms needed
to locate PROFS document

3 COMPLETED SEARCH FOR DOCUMENTS D03
Look at list of documents found
Press PF1

4 LIST OF THE DOCUMENTS FOUND D04
Choose document from list and
press corresponding PF key

5 PROCESS THE DOCUMENT FOUND D08
Press PF10 to
Look at the next screen

6 PROCESS THE DOCUMENT FOUND D02
Press PF2 to Copy the document
into your personal storage

7 PROFESSIONAL OFFICE SYSTEM A00
Keep pressing PF12 until back to
PROFS main menu – then press PF5

8 PREPARE DOCUMENTS F00
Enter filename that was
placed on A-disk and press PF3

9 PROCESS THE DOCUMENT F01
Press PF5 to file and send the
document as a Final document

10 SEND THE FINAL DOCUMENT F06
Press PF2 to erase the SELECTED
parameter and press ENTER

11 PROFESSIONAL OFFICE SYSTEM A00
Keep pressing PF12 until back to
PROFS main menu

Notes:

After step 6, assuming the document is in "final" form, you will receive a message saying "'Dxxxxxxx MEMO' has been placed in your personal storage," where 'Dxxxxxxx' is a number such as: D2890001. If the document was in "draft" form, the message will read: "'Dxxxxxxx SCRIPT' has been placed in your personal storage." In this case, the document must be converted to "final" form, as described in "Changing a DCF file to 1403 Format" on page 60, ready to be sent to DISOSS.

5.4 SENDING A PROFS NOTE TO DISOSS

A PROFS note is an unformatted file, containing no DCF control words or tags.

A note can be transferred from the PROFS notelog, which is on your A-disk, by sending the note to your own CMS userid. PROFS will place the note in your virtual reader, from where it can be read in via RDRLIST and edited as necessary and transferred to DISOSS by the "DBTNOTE" EXEC.

5.4.1 FORWARDING AN EXISTING NOTE

We used the following steps to send a PROFS note to DISOSS/PS:

1. From the PROFS main menu (A00), press **PF4**, Process notes and messages.

2. From the PROCESS NOTES AND MESSAGES menu (E05), press **PF3**, Look at the **Note** Log.

3. From the LOOK AT THE NOTE LOG menu (E08), select a note to be sent to DISOSS, and press the corresponding PF key.

4. From the PROCESS THE NOTE LOG menu (E09), press **PF5**, Forward the note.

5. From the FORWARD THE NOTE menu (E11), enter your own CMS userid after "Forward to:", and press **PF7**.

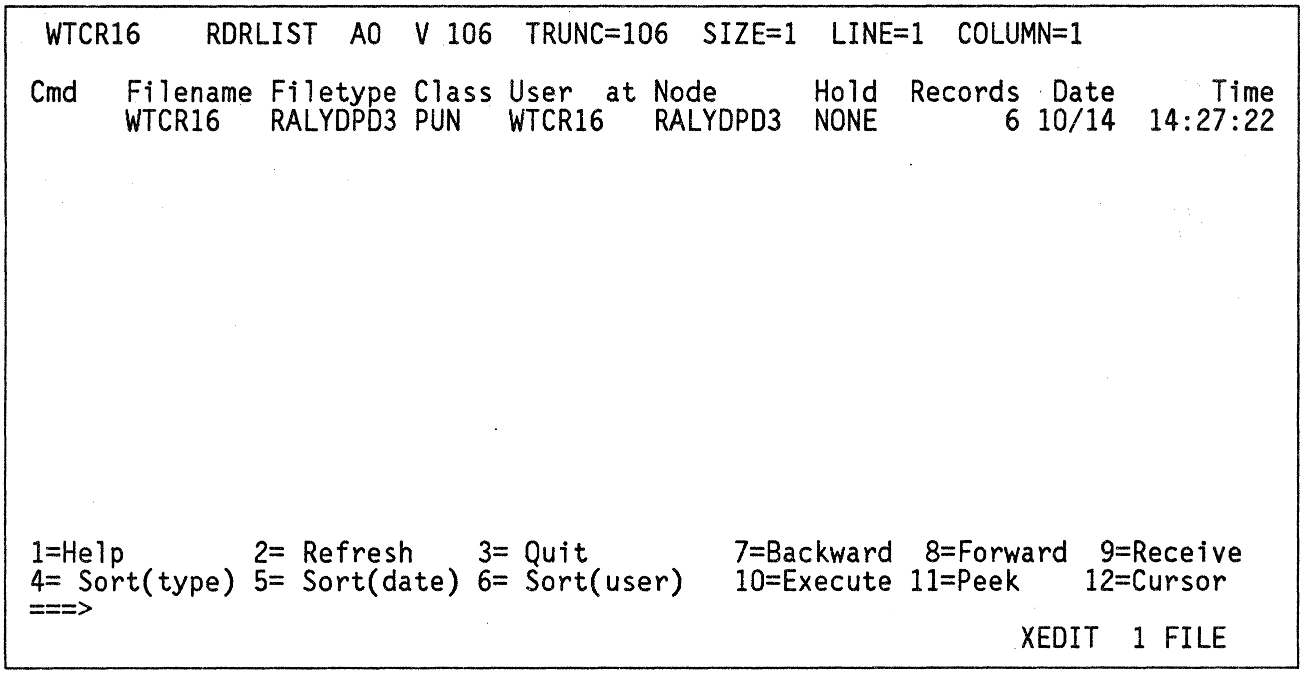
   You will receive a message saying:

   ```
   PUN FILE xxxx TO userid COPY 001  NOHOLD
   OFSNSP002I SENT TO <userid> AT <nodeid>
   ```

6. Press **PF12** until back to the PROFS main menu.

7. Press the PF key for "DISOSS Tasks" (in our case, **PF10**), then select Option 3, "Read in and send a Note to DISOSS Users", and press ENTER.

When the RDRLIST screen appears, enter the command **DBTNOTE** under "Cmd", alongside the PUN file created by the above steps and press PF10.



```
 WTCR16    RDRLIST  A0  V 106  TRUNC=106  SIZE=1  LINE=1  COLUMN=1

Cmd   Filename Filetype Class User  at Node      Hold  Records  Date      Time
      WTCR16   RALYDPD3 PUN   WTCR16   RALYDPD3  NONE        6 10/14  14:27:22













1=Help       2= Refresh    3= Quit          7=Backward  8=Forward  9=Receive
4= Sort(type) 5= Sort(date) 6= Sort(user)   10=Execute 11=Peek    12=Cursor
===>
                                                           XEDIT  1 FILE
```

5.4.2 THE DBTNOTE EXEC

An example of the DBTNOTE EXEC is shown in "Sample DBTNOTE EXEC" on page 163.

DBTNOTE edits the note file that may look like the following example:

```
==============================================================================
 MSG:FROM: WTCR16  --RALYDPD3 TO: WTCR16  --RALYDPD3         10/12/83 11:58:12
 To: WTCR16  --RALYDPD3

 Subject: Forwarding Note 08/05/83 14:57 Sending Notes
                    * * * F O R W A R D E D   N O T E * * *
 To: MILLAR  --RALYDPD3                    MILLAR  --RALYDPD3
 Subject: Sending Notes
 John:
 This is a test note being sent from SYSTEM3(WTCR16) to your signon.

 Cheers...  Joe Bloggs
      Forwarding Note 08/05/83 14:57 Sending Notes
```

For this example, the output from DBTNOTE would be:

```
 Subject: Sending Notes
 John:
 This is a test note being sent from SYSTEM3(WTCR16) to your signon.

 Cheers...  Joe Bloggs
```

This is then submitted within an MVS job via RJE, to become an entry in the DISOSS/PS mail-log.

5.4.3 CREATING AND SENDING A NEW NOTE

This procedure is similar to the previous one, with the exception of the following steps:

1. From the PROCESS NOTES AND MESSAGES menu (E05), press **PF1** Send a note.
2. Enter your own CMS userid or nickname after "**Send to:**", fill in the note and press **PF7** (Send).

From here, the procedure is the same as before.

5.5 CHANGING A DCF FILE TO 1403 FORMAT

When a "final" PROFS document is retrieved from the database and placed on the A-disk, it will be in 1403 format. If you retrieve a "draft" document, it will be in DCF input format, that is, the document will consist of the text, SCRIPT control words and tags. The document must be converted to 1403 format before it can be transferred to DISOSS/PS.

You could SCRIPT the document from CMS using the PROFS starter set Profile OFSMPROF instead of the normal DCF Release 2 Profile - SSPROF. This entails some extra work, as PROFS passes tokens to the $FORMAT EXEC on the SYSADMIN 399 disk when the document is processed normally by PROFS.

We chose to use existing PROFS facilities to process a draft document by following the steps listed below: This means there are a few extra steps involved, but if PROFS is changed to a newer release, the method is still valid, whereas a customer written EXEC runs the risk of needing to be re-written to work with the new release.

1. From the PROFS main menu (A00) press **PF3** Search for documents.

2. From the SEARCH FOR DOCUMENTS menu (D01), enter the search terms needed to retrieve the draft document.

3. From the COMPLETED SEARCH FOR DOCUMENTS menu (D03), press **PF1** Look at list of documents found with the mail log comments.

4. From the LIST OF DOCUMENTS FOUND menu (D04), press the corresponding PF key to select the draft document to be copied into the database as a final document. The original draft document will remain as before.

5. From the PROCESS THE DOCUMENT FOUND menu (D11), press **PF10** Look at the next screen.

6. From the next PROCESS THE DOCUMENT FOUND menu (D09), press **PF2** Copy the document into your personal storage.

   You will receive a message saying "'Dxxxxxxx SCRIPT' has been placed in your personal storage," where 'Dxxxxxxx ' is a number such as: D2890001. Note this filename for use in next steps.

7. Press **PF12** until you get back to the main menu. From the main menu, press **PF5** Prepare Documents.

8. From the Prepare Documents menu (F00), enter the filename (noted in step 6), after **PF3** Change a Draft Document. Then press **PF3**.

9. From the Process the document menu (F01), press PF5 File and send the document as a Final document.

10. From the Send the Final document menu (F06), press PF2 to erase the SELECTED parameter, and press ENTER.

    You will receive a message saying "DOCUMENT ASSIGNED 83xxxxxxxx". Press CLEAR. You are now back at the PROFS main menu (D03), and a copy of the draft document is now stored in the data base in final form. This final form document can now be retrieved and transferred to DISOSS/PS as described in "Sending a PROFS document to DISOSS" on page 55.

5.6 LOADING A DISOSS/PS DOCUMENT TO THE A-DISK AND THEN TO PROFS

The DISOSS/PS document is transferred to CMS as a print file in the virtual reader. From here it can be processed in two ways:

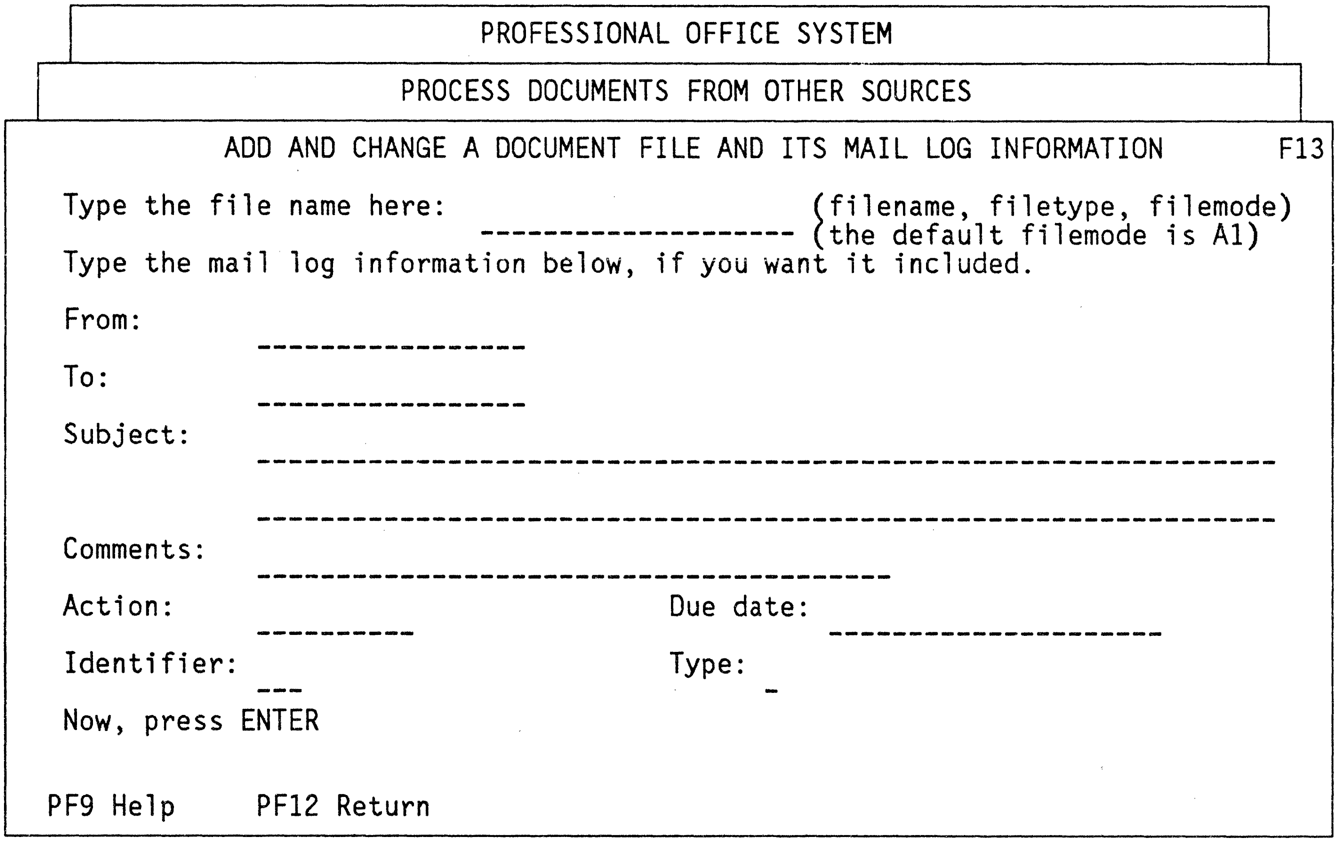
- Via Option 4 of the DBTMENU menu, which invokes RDRLIST and allows the DBTRECV command to be issued.

- Via the PROFS 'Open the Mail' function.

Then you can browse or edit the document, and note any information you may need when describing the document to PROFS later.

From the PROFS main menu, press the PF key to Process documents from other sources (usually PF6).

From the PROCESS DOCUMENTS FROM OTHER SOURCES menu, press **PF2 Add and change a document file and its mail log information.**

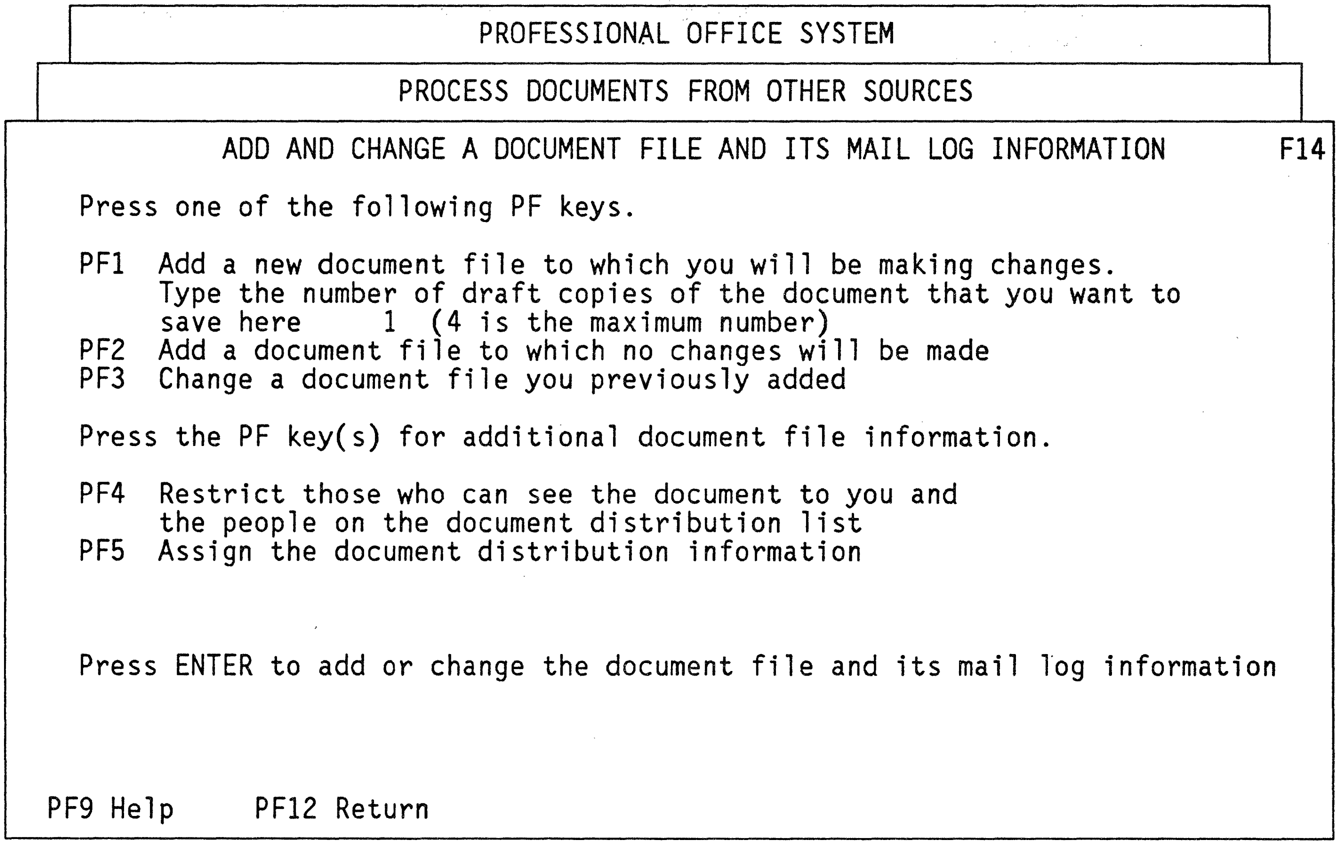


```
                        PROFESSIONAL OFFICE SYSTEM

                    PROCESS DOCUMENTS FROM OTHER SOURCES

          ADD AND CHANGE A DOCUMENT FILE AND ITS MAIL LOG INFORMATION       F13

   Type the file name here:                         (filename, filetype, filemode)
                            -------------------- (the default filemode is A1)
   Type the mail log information below, if you want it included.

   From:
               -----------------
   To:
               -----------------
   Subject:
               -----------------------------------------------------------------

               -----------------------------------------------------------------
   Comments:
               ----------------------------------------
   Action:                          Due date:
               ----------                     ---------------------
   Identifier:                      Type:
               ---                              -
   Now, press ENTER


  PF9 Help     PF12 Return
```

Figure 13. PROFS menu to Add/Change a Document File

The cursor is positioned for you to enter file identifier information (file name, file type, and file mode).

When you press ENTER, you will see a second ADD AND CHANGE A DOCUMENT FILE AND ITS MAIL LOG INFORMATION menu.



```
                         PROFESSIONAL OFFICE SYSTEM
                     PROCESS DOCUMENTS FROM OTHER SOURCES
             ADD AND CHANGE A DOCUMENT FILE AND ITS MAIL LOG INFORMATION       F14

    Press one of the following PF keys.

    PF1  Add a new document file to which you will be making changes.
         Type the number of draft copies of the document that you want to
         save here     1  (4 is the maximum number)
    PF2  Add a document file to which no changes will be made
    PF3  Change a document file you previously added

    Press the PF key(s) for additional document file information.

    PF4  Restrict those who can see the document to you and
         the people on the document distribution list
    PF5  Assign the document distribution information



    Press ENTER to add or change the document file and its mail log information




  PF9 Help      PF12 Return
```

This menu requests information about how you want the document file stored.

1. If you are filing a document for the first time and plan to change it later, use **PF1 Add a new document file to which you will be making changes.** With this choice, you can also specify the number of versions of this file that you want stored as it is revised. The system will save one version if you do not change the number on this menu. The maximum number of versions you can save is four.

2. Use **PF2 Add a document file to which no changes will be made** if you are storing it as a final document.

3. Use **PF3 Change a document file you previously stored** if you are storing a changed copy of a document that is already in the central file.

4. Use **PF4** to **restrict** access to the document.

5. Use **PF5 Assign the document distribution information** to forward the document to other people.

Press ENTER when you finish making selections.

After you have finished, you will see the assigned document number and the message:

```
                    DELETE FROM PRIVATE WORKSPACE?
```

1. If you type "y" or "yes" and press the ENTER key, the original document file will be erased. Access to the document will only be possible through PROFS.

2. If you type "n" or "no" and press ENTER, the original copy will remain on your personal storage.

APPENDICES

A.0 SAMPLE SYSTEM COMPONENTS

A.1 GENERAL REMARKS

This section contains the code used in the course of the project to verify the design. These examples could form a basis for implementation at another location, but it should be clearly recognised that they were never intended to be anything other than demonstration code. They lack detailed documentation, adequate error notification and recovery, and professional programmers will find them inefficiently and unimaginatively coded.

During the early stages of the development of these programs, an error in our PL/I compiler involving concatenation to varying strings required fixed strings to be used in places where varying strings may seem more appropriate. The error was later corrected, but it influenced the coding of PL/I programs DBTBAT1 and DBTTRN1.

The following were set up for the CICS PL/I programs:

- A partitioned dataset containing the control blocks DBTVSQ, and DBTOC, and the six DIU build subroutine control blocks listed in Appendix A, is included in the SYSLIB DD statement of the compile step of the CICS procedure CICEITPL. The INCLUDE option must be among the compile step PARM options in order for these blocks to be included.
- The library containing the load modules of the DIU routines is included in the linkedit step of the CICEITPL procedure, in order for these routines to be linked into the CICS PL/I programs.

A.2 SOFTWARE USED TO TEST THE DESIGN

The systems used to test the design were:

- MVS/SP-JES2 Version 1 Release 3.3
- CICS/OS/VS Release 1.6.1
- DISOSS/370 Version 3 Release 1
- DISOSS/Professional Support Release 1

A.3 SOURCE LISTINGS

Below is a summary of the common blocks, programs and JCL used by the Box X to DISOSS (Inbound) function.

DBTVSQ	Common block for VSAM record input/output overlays.
DBTOC	Subroutine communication field for open and close macro.
DBTDOCIN	Catalogued procedure invoked by batch job.
DBTBAT1	PL/I program to place document on DBTVSQ0.
DBTMOV1	PL/I program to move request from DBTVSQ0 to DBTVSQ1.
DBTOPN1	Assembler program to invoke DFHOC open for DBTMOV1.
DBTCLS1	Assembler program to invoke DFHOC close for DBTMOV1.
DBTMST1	PL/I program to select appropriate transform program.
DBTTRN1	PL/I program to transform request text.
DBTSND1	PL/I program to send request to API.
DBTRSP1	PL/I program to process responses received from DISOSS.
DBTSON1	PL/I program to signon to DISOSS.
DBTCLN1	PL/I program to reset VSAM datasets.
DBTSOF1	PL/I program to signoff from DISOSS.
DBTTRT01	Job to assemble and link-edit a DISOSS translate table.

A.3.1 DBTVSQ COMMON BLOCK

DBTVSQ is a common block which defines the input and output record structures for files DBTVSQ0 and DBTVSQ1.

The 3rd level fields are intended for reference to specific parameters. The 2nd level fields are intended for larger scale manipulation of sets of parameters.

```
/********************************************************************/
/*                                                                  */
/* DBTVSQ: VSQ0/VSQ1 I/O RECORD OVERLAYS                            */
/*    THE INPUT & OUTPUT AREA POINTERS, VIPTR & VOPTR ARE DEFINED   */
/* IN EACH APPLICATION PROGRAMS CONTROL BLOCK, TO FACILITATE        */
/* DEBUGGING                                                        */
/********************************************************************/
 DCL 1 VI CHAR(6002) BASED(VIPTR), /* INPUT MAP                     */
     1 VIG BASED(VIPTR), /* 2ND LEVEL FIELDS                        */
       2 VIKEY CHAR(39),
       2 VIDATA CHAR(5961) VARYING,
     1 VID BASED(VIPTR), /* 3RD LEVEL FIELDS                        */
       2 VI3KEY,
         3 DATE CHAR(5),        /* DATE                             */
         3 TIME CHAR(9),        /* TIME                             */
         3 INTTYPE CHAR(1),     /* INTERFACE TYPE                   */
         3 OSN CHAR(8),         /* OSN NAME                         */
         3 USER CHAR(8),        /* USER NAME FOR DISTRIBUTION       */
         3 INTYPE BIT(16),      /* INPUT DOCTYPE DIA-CODED          */
         3 OUTYPE BIT(16),      /* OUPUT DOCTYPE DIA-CODED          */
         3 SEQNO FIXED BIN(15), /* SEQUENCE NO X'0000' TO X'FFFF'   */
         3 CHFLAGF CHAR(1),     /* CHAINING FLAG 1                  */
         3 CHFLAGL CHAR(1),     /* CHAINING FLAG 2                  */
       2 VI3DATA,         /* PROFILE FIELDS FOR FIC/LIC/OIC         */
         3 DUMLEN  BIN(15) FIXED, /* DUMMY LENGTH FIELD */
         3 RECTYP  CHAR(1),  /* RECORD TYPE A ->                    */
         3 PROFLAG CHAR(1),  /* RECORD SIZE IN CARD IMAGE UNITS     */
         3 PAGEL   CHAR(3),  /* PAGELENGTH IN LINES/PAGE            */
         3 PAGEW   CHAR(3),  /* PAGEWIDTH IN CHAR/LINE              */
         3 DISNAM  CHAR(8),  /* DISTRIBUTION ID                     */
         3 EYECAT  CHAR(6),  /* 'HEADER' CONSTANT                   */
         3 DOCNAM  CHAR(15), /* DOCUMENT NAME                       */
         3 DISFIL  CHAR(1),  /* DISTRIBUTE OR FILE                  */
         3 RESER   CHAR(42), /* RESERVED                            */
     1 VIC BASED(VIPTR),             /*GENERIC KEY                  */
       2 VICKEY CHAR(35),            /*GENERIC KEY                  */
       2 VICKEY2 CHAR(4),            /*KEY LAST PART                */
       2 VICDUM CHAR(2),             /*DUMMY LENGTH FIELD           */
       2 VICHEAD CHAR(80),           /* HEADER                      */
     1 VIDKEY BIT(312) BASED(VIPTR); /*KEY IN BIT                   */
 DCL 1 VO CHAR(6002) BASED(VOPTR),   /* OUTPUT MAP                  */
     1 VOG BASED(VOPTR), /* 2ND LEVEL FIELDS                        */
       2 VOKEY CHAR(39),
       2 VODATA CHAR(5961) VARYING,
     1 VOD BASED(VOPTR), /* 3RD LEVEL FIELDS                        */
       2 VO3KEY,
         3 DATE CHAR(5),        /* DATE                             */
         3 TIME CHAR(9),        /* TIME                             */
         3 INTTYPE CHAR(1),     /* INTERFACE TYPE                   */
         3 OSN CHAR(8),         /* OSN NAME                         */
```

```
    3 USER CHAR(8),        /* USER NAME FOR DISTRIBUTION       */
    3 INTYPE BIT(16),      /* INPUT DOCTYPE DIA-CODED          */
    3 OUTYPE BIT(16),      /* OUPUT DOCTYPE DIA-CODED          */
    3 SEQNO FIXED BIN(15), /* SEQUENCE NO X'0000' TO X'FFFF'   */
    3 CHFLAGF CHAR(1),     /* CHAINING FLAG 1                  */
    3 CHFLAGL CHAR(1),     /* CHAINING FLAG 2                  */
  2 VO3DATA,         /* PROFILE FIELDS FOR FIC/LIC/OIC         */
    3 DUMLEN  BIN(15) FIXED, /* DUMMY LENGTH FIELD */
    3 RECTYP  CHAR(1),  /* RECORD TYPE A ->                    */
    3 PROFLAG CHAR(1),  /* RECORD SIZE IN CARD IMAGE UNITS     */
    3 PAGEL   CHAR(3),  /* PAGELENGTH IN LINES/PAGE            */
    3 PAGEW   CHAR(3),  /* PAGEWIDTH IN CHAR/LINE              */
    3 DISNAM  CHAR(8),  /* DISTRIBUTION ID                     */
    3 EYECAT  CHAR(6),  /* 'HEADER' CONSTANT                   */
    3 DOCNAM  CHAR(15), /* DOCUMENT NAME                       */
    3 DISFIL  CHAR(1),  /* DISTRIBUTE OR FILE                  */
    3 RESER   CHAR(42); /* RESERVED                            */
```

A.3.2 DBTOC COMMON BLOCK

```
/****************************************************/
/*                                                  */
/*  DBTOC:                                          */
/*    SUBROUTINE COMMUNICATION FIELD                */
/*            FOR OPEN AND CLOSE MACRO.             */
/****************************************************/
 DCL OPENPTR POINTER;
 DCL
   1 OPENBLK BASED(OPENPTR),
     2 DBNAME CHAR(8),
     2 RC     BIT(8),
     2 FCT    BIT(24),
     2 FFF    BIT(24);
/****************************************************/
/*                                                  */
/*   DBTMOV1  CONTROL BLOCK                         */
/*            USED FOR REMEMBERING FIELDS ETC       */
/****************************************************/
 DCL Q0Q1PTR POINTER;
 DCL
   1 Q0Q1    BASED(Q0Q1PTR),
   2 KEY,               /* KEY 39 CHARACTERS                        */
     3 DATE CHAR(5),        /* DATE                                 */
     3 TIME CHAR(9),        /* TIME                                 */
     3 INTTYPE CHAR(1),     /* INTERFACE TYPE                       */
     3 OSN CHAR(8),         /* OSN NAME                             */
     3 USER CHAR(8),        /* USER NAME FOR DISTRIBUTION           */
     3 INTYPE BIT(16),      /* INPUT DOCTYPE DIA-CODED              */
     3 OUTYPE BIT(16),      /* OUPUT DOCTYPE DIA-CODED              */
     3 SEQNO FIXED BIN(15), /* SEQUENCE NUMBER X'0000' TO X'FFFF'   */
     3 CHFLAGF CHAR(1),     /* CHAINING FLAG 1                      */
     3 CHFLAGL CHAR(1),     /* CHAINING FLAG 2                      */

   2 HEAD,                  /* ONLY FOR FIC OR LIC                  */
      3 RECTYP    CHAR(1),  /* RECORD TYPE A ->                     */
      3 PROFLAG   CHAR(1),  /* RECORD SIZE IN CARD IMAGE UNITS      */
      3 PAGEL FIXED DEC(3,0), /* DECIMAL PAGELENGTH IN LINES/PAGE   */
      3 PAGEW FIXED DEC(3,0), /* DECIMAL PAGEWIDTH IN CHAR/LINE     */
      3 DISNAM CHAR(8),     /* UNUSED                               */
      3 EYECAT    CHAR(6),  /* 'HEADER' CONSTANT                    */
      3 DOCNAM CHAR(15),    /* DOCUMENT NAME                        */
      3 RESER CHAR(43),     /* RESERVED FILED LENGHT                */
    2 COUNTER1 FIXED BIN(15,0),
    2 COUNTER2 FIXED BIN(15,0),
    2 VIPTR POINTER,

 1 Q0Q1B BASED(Q0Q1PTR),   /*FIC OR OIC OR LIC                      */

   2 KEY,               /* KEY 39 CHARACTERS                        */
     3 KEY1 CHAR(35),      /* FIXED PART FOR ONE DOC                */
     3 KEY2 CHAR(4),        /* USER NAME FOR DISTRIBUTION           */

   2 HEAD CHAR(80),         /* ONLY FOR FIC OR LIC                  */

 1 Q0Q1C BASED(Q0Q1PTR),   /*FIC OR OIC OR LIC                      */

   2 KEY CHAR(39);      /* KEY 39 CHARACTERS                        */
```

A.3.3 DBTDOCIN CATALOGUED PROCEDURE

DBTDOCIN is a sample catalogued procedure invoked by the batch job submitted from Box X.

```
//DBTDOCIN PROC
//*
//***********************************************************
//*                                                         *
//* DISOSS - BATCH INTERFACE.                               *
//*                                                         *
//* THIS PROCEDURE IS INVOKED BY A BATCH JOB SUBMITTED BY   *
//* 'BOX X'. FUNCTION IS TO REBUILD 1403 PRINTLINES FROM    *
//* INPUT CARD IMAGES, CONCATENATE THEM IN LARGE PHYSICAL   *
//* RECORDS, AND INSERT THEM IN THE BATCH-->CICS INTERFACE  *
//* DATASET DBTVSQ0.                                        *
//*                                                         *
//***********************************************************
//*
//INSERT  EXEC PGM=DBTBAT1
//STEPLIB   DD DSN=DISOSS30.DBT.LOADLIB,DISP=SHR
//          DD DSN=F5.PLIBASE,DISP=SHR
//SYSPRINT  DD SYSOUT=A
//LOGFILE   DD SYSOUT=A
//BUFFER    DD DCB=(RECFM=V,LRECL=6006),SPACE=(CYL,(1,1)),UNIT=SYSDA
//VSQ0      DD DSN=DISOSS30.DBTVSQ0,DISP=SHR
//*
//* INPUT DDNAME IS DOCIN
//*
```

A.3.4 DBTBAT1 PROGRAM SOURCE

This program uses its own DBTVSQ1 output record definitions. The reasons for this are historical. There is no reason why they should not now be changed to conform with DBTVSQ.

If the dataset DBTVSQ0 is open to the CICS transaction DBTM, the program will repeatedly branch to the label OPENFILE to repeat its open attempts until successful. The use of a loop to enforce a delay period between open attempts is unsatisfactory because it consumes considerable processing resource. It could be replaced with an assembler subroutine to issue a WAIT macro.

The sample program does nothing to avoid two batch jobs using the PL/I TIME pseudovariable during the same millisecond, which could cause duplicate keys on the VSAM datasets. This problem can be solved in the following way. A recursive on-ILLOGIC branch to a routine could be set up before the FIC write. The routine could either add one to VSQD.KEY.TIME, or could reassign the TIME psudovariable. Continued attempts would then be made until a unique key was found. Alternatively, an assembler subroutine could issue an ENQUEUE before getting the date and time.

The record of error codes in the file LOGFILE was found genuinely useful during debugging, although it was intended merely as an example of a way to begin error detection.

```
DBTBAT1:  PROC OPTIONS(MAIN);
 /********************************************************/
 /*                                                      */
 /* DBTBAT1   15/09/1983                                 */
 /*    PL1 PROGRAM SOURCE DBTBAT1 FOR PROCEDURE DBTDOCIN */
 /*    CALLED BY BATCH JOB SUBMITTED FROM BOX X          */
 /*                                                      */
 /* INPUT:  JCL INSTREAM DATA                            */
 /*    1) HEADER WITH USER, PROFILE, AND PROCESSING DATA */
 /*    2) CHOPPED UP PRINT LINES                         */
 /*                                                      */
 /* OUTPUT: KEY SEQUENCED 6K VSAM RECORDS TO DBTVSQ0     */
 /*    1) KEYS WITH USER AND PROCESSING DATA             */
 /*    2) PROFILE DATA ON FIRST AND LAST RECORD          */
 /*    3) IRS SEPARATED PRINTLINES                       */
 /*                                                      */
 /* OUTBOUND RECORDS ARE FIRST WRITTEN TO A BUFFER, AND  */
 /* THEN COPIED TO DBTVSQ0, TO MINIMISE THE PERIOD OF    */
 /* TIME DURING WHICH DBTVSQ0 IS HELD OPEN               */
 /*                                                      */
 /********************************************************/
  DCL (DATE,TIME,MIN,SUBSTR,UNSPEC,LENGTH,MAX,ADDR) BUILTIN;
 /*------------------- DATASET DEFINITIONS -----------------*/
  DCL DOCIN   FILE RECORD, /* INPUT FILE                    */
      BUFFER  FILE RECORD, /* TEMPORY BUFFER FILE           */
      LOGFILE FILE RECORD, /* ERROR LOGGING FILE            */
      VSQ0    FILE RECORD OUTPUT DIRECT BUFFERED KEYED
                  ENV(VSAM); /* OUTPUT FILE                 */
 /* THE DCB FOR BUFFER IS DCB=(RECFM=V,LRECL=6006)          */
 /* THE DCB FOR LOGFILE IS DCB=RECFM=F,LRECL=80,BLKSIZE=80) */
 /*------------- DOCIN HEADER RECORD LAYOUT ---------------*/
  DCL 1 HEADER,
          3 HOSN      CHAR(8), /* OSN NAME */
```

```
           3 HUSER     CHAR(8), /* USER NAME FOR DISTRIBUTION */
           3 HINTYPE   CHAR(2), /* INPUT DOCUMENT TYPE */
           3 HOUTYPE   CHAR(2), /* OUTPUT DOCUMENT TYPE */
           3 HRECSIZE  CHAR(1), /* NUMBER OF 80-BYTE CARD */
                                /* IMAGES PER PRINT LINE */
           3 HRECTYPE  CHAR(1), /* RECORD TYPE */
           3 HPROFLAG  CHAR(1), /* PROFILE FLAG */
           3 HPAGEL    CHAR(3), /* PAGELENGTH IN LINES/PAGE */
           3 HPAGEW    CHAR(3), /* PAGEWIDTH IN CHARS/LINE */
           3 HDISNAM   CHAR(8), /* RESERVED */
           3 HEYECAT   CHAR(6), /* 'HEADER' CONSTANT */
           3 HDOCNAM  CHAR(15), /* DOCUMENT NAME */
           3 HDISFIL  CHAR(1),  /* FILE OR DISTRIBUTE */
           3 HRESER    CHAR(21); /* RESERVED */
/*----------VSQO RECORD OVERLAYS --------------------*/
  DCL VSQPTR POINTER;
  DCL VSQBASE BASED(VSQPTR) CHAR(6000), /* BASIC OVERLAY AREA       */
   1 VSQD  BASED(VSQPTR),   /* LEVEL 3 FIELDS                         */
    2 KEY,                  /* KEY                                    */
     3 DATE    CHAR(5),     /* DATE                                   */
     3 TIME    CHAR(9),     /* TIME                                   */
     3 INTTYPE CHAR(1),     /* INTERFACE TYPE                         */
     3 OSN     CHAR(8),     /* OSN NAME                               */
     3 USER    CHAR(8),     /* USER NAME FOR DISTRIBUTION             */
     3 INTYPE  BIT(16),     /* INPUT DOCTYPE DIA-CODED                */
     3 OUTYPE  BIT(16),     /* OUPUT DOCTYPE DIA-CODED                */
     3 SEQNO FIXED BIN(15), /* SEQUENCE NUMBER X'0000' TO X'FFFF'     */
     3 CHFLAGF CHAR(1) INIT('1'), /* CHAINING FLAG 1                  */
     3 CHFLAGL CHAR(1) INIT('0'), /* CHAINING FLAG 2                  */
    2 DUMDATA  CHAR(5961),  /* DUMMY DATA FIELD                       */
   1 VSQA BASED(VSQPTR),    /* LEVEL 2 FIELDS                         */
    2 KEY CHAR(39),         /* KEY                                    */
    2 DATA CHAR(5959) VARYING; /* RECORD DATA                         */
    DCL LENPOINT POINTER,
        LENBIN FIXED BIN(15,0) INIT(1) BASED(LENPOINT),
        LENCHAR      CHAR(2)           BASED(LENPOINT);
/*------------------- PROFILE OVERLAYS ---------------*/
  DCL PRPTR POINTER; /* PROFILE OVERLAY POINTER */
  DCL 1 PROFILED BASED(PRPTR),  /* PROFILE DETAIL                    */
     3 RECTYP    CHAR(1),   /* RECORD TYPE                            */
     3 PROFLAG   CHAR(1),   /* RECORD SIZE IN CARD IMAGE UNITS        */
     3 PAGEL     CHAR(3),   /*NOT DECIMAL PAGELENGTH IN LINES/PAGE    */
     3 PAGEW     CHAR(3),   /*NOT DECIMAL PAGEWIDTH IN CHAR/LINE      */
     3 DISNAM CHAR(8),      /* UNUSED                                 */
     3 EYECAT    CHAR(6),   /* 'HEADER' CONSTANT                      */
     3 DOCNAM CHAR(15),     /* DOCUMENT NAME                          */
     3 DISFIL CHAR(1),      /* DISTRIBUTE OR FILE                     */
     3 RESER CHAR(42),      /* RESERVED FILED LENGHT                  */
    PROFILEA CHAR(80) BASED(PRPTR);    /* PROFILE GENERAL AREA        */
/*------------------- VARIABLES ----------------------*/
  DCL CARD     CHAR(80), /* 80-BYTE I/O CARD IMAGE          */
      VSR      CHAR(6002) VARYING, /* VARYING OUTPUT STRING  */
      PL       CHAR(500) VARYING, /* PRINT LINE             */
      PLPTR    BIN(15,0)  FIXED, /* PRINT LINE POINTER      */
      ERROR    DEC(4)     FIXED INIT(0), /* ERROR CODE      */
      NL       CHAR(1), /* NEWLINE CHARACTER                */
      NLBP     POINTER,/* NEWLINE BIT OVERLAY POINTER      */
      NLB      BIT(8)     BASED(NLBP) INIT('00011110'B),
                        /* NEWLINE BIT OVERLAY              */
      N        DEC(7)     FIXED INIT(0), /* SCRATCH VARIABLE */
```

```
       N1       DEC(7)     FIXED INIT(0); /* SCRATCH VARIABLE */
 /*------------- SET UP -------------------------------*/
       ON ENDFILE (DOCIN)  GOTO FINAL;
       ON ENDFILE (BUFFER) GOTO STOP;
       ON RECORD  (BUFFER) GOTO CONT1;
       ON UNDEFINEDFILE(VSQ0) GOTO OPENFILE;
       ALLOCATE NLB;
       ALLOCATE VSQBASE;
       ALLOCATE PROFILED;
       ALLOCATE LENBIN;
       NLBP = ADDR(NL);
       NLB = '00011110'B;
 /*------------- CONSTRUCT VSAM HEADER ----------------*/
 /*****************************************************/
 /*    THE MINIMUM OF PROCESSING IS PERFORMED HERE.      */
 /*    INPUT PARAMETERS WHICH CAN HAVE ONLY ONE          */
 /*    MEANINGFUL VALUE ARE IGNORED                      */
 /*****************************************************/
       KEY.DATE=SUBSTR(DATE,2);
       KEY.TIME=TIME;
       KEY.INTTYPE='A';
       READ FILE (DOCIN) INTO (HEADER);
       OSN = HOSN;
       IF OSN = ' ' THEN OSN = 'DSVHOST ';
       USER = HUSER;
       IF USER = ' ' THEN USER = 'HDVF02  ';
       INTYPE = '0000000000001100'B; /* TO BE DEP ON INTYPES */
       OUTYPE = '0000000000000010'B; /* TO BE DEP ON OUTYPES */
       RECTYP = 'A'; /* TO BE DEPENDENT ON RECTYPES */
       PROFLAG = 'A'; /* TO BE DEPENDENT ON PROFLAGS */
       PAGEL = HPAGEL;
       PAGEW = HPAGEW;
       DISNAM = HDISNAM;
       IF DISNAM = ' ' THEN DISNAM = USER;
       EYECAT = HEYECAT;
       IF EYECAT ¬= 'HEADER' THEN GOTO ERROR1;
       DOCNAM = HDOCNAM;
       DISFIL = HDISFIL;
       CHFLAGF = '1';
       CHFLAGL = '0';
       IF HPROFLAG = 'A' THEN GOTO TEXT;
 /*-----------PROCESS PROFILE TYPE 'B' DATA -----------*/
 /*                                                    */
 /*                                                    */
 /*          (TO BE CONTINUED ....    )                */
 /*                                                    */
 /*                                                    */
 /*---------------- EXTRACT NEXT LINE -----------------*/
TEXT:   PL = '';
        N = UNSPEC(HRECSIZE) - 15*16;
        DO N1 = 1 TO N;
        READ FILE (DOCIN) INTO (CARD);
        PL = PL || CARD;
        END;   /* N1 */
        PLPTR = LENGTH(PL);
 /*---------------- REMOVE TRAILING BLANKS ------------*/
NEXTCHAR: IF SUBSTR(PL,PLPTR,1)¬=' ' | PLPTR=1 THEN GOTO REMOVE;
        PLPTR = PLPTR - 1;
        GOTO NEXTCHAR;
 /*---------------- REMOVE UNPRINTABLE CHARACTERS -----*/
```

```
REMOVE:   PL = SUBSTR(PL,1,PLPTR);
          N1 = PLPTR + 1;
PREVCHAR: N1 = N1 - 1;
          IF N1 < 2 THEN GOTO ADDLINE;
          IF UNSPEC(SUBSTR(PL,N1,1)) < 64 THEN
             PL = SUBSTR(PL,1,N1-1) || ' ' || SUBSTR(PL,N1+1);
          GOTO PREVCHAR;
 /*---------------- TEST FOR FULL OUTPUT RECORD -------*/
ADDLINE:  IF LENGTH(DATA) + LENGTH(PL)
                < 6000-39-80-2+1 THEN GOTO SAMEREC;
 /*----------- INCLUDE PROFILE DATA IF FIC/LIC/OIC ----*/
LASTONE:  IF SEQNO=0 | CHFLAGF='1' | CHFLAGL='1' THEN
             DATA = PROFILEA || DATA;
 /*---------------- WRITE OUTPUT RECORD ----------------------*/
          LENBIN = LENGTH(VSQA.DATA);
          VSR = VSQA.KEY || LENCHAR || VSQA.DATA;
          WRITE FILE (BUFFER)  FROM (VSR);
CONT1:    IF CHFLAGL = '1' THEN GOTO COPY;
          SEQNO = SEQNO + 1;
          CHFLAGF = '0';
          DATA = '';
 /*---------- APPEND PRINT LINE TO RECORD UNDER CONSTRUCTION -*/
SAMEREC:  DATA = DATA || PL || NL;
          GOTO TEXT;
 /*------- REPEAT ATTEMPTS TO OPEN VSQO UNTIL SUCCESSFUL ---*/
OPENFILE: DO N = 1 TO 999;
          CARD = 'THIS IS AN UNSATISFACTORY WAY OF WAITING';
          END;
          OPEN FILE(VSQO) OUTPUT;
          GOTO AGAIN;
 /*---SET CHAINFLAG LAST TO REMEMBER TO STOP AFTER NEXT WRITE -*/
FINAL:  CHFLAGL = '1';
        GOTO LASTONE;
 /*------------ COPY BUFFER INTO REQUEST QUEUE --------*/
COPY:   CLOSE FILE(BUFFER);
        ON RECORD(BUFFER) GOTO CONT2;
        OPEN FILE(BUFFER) INPUT;
        OPEN FILE(VSQO)  OUTPUT;
AGAIN:  READ  FILE(BUFFER)  INTO (VSR);
CONT2:  IF LENGTH(VSR) < 41 THEN GOTO ERROR3;
        VSQA.KEY = SUBSTR(VSR,1,39);
        WRITE FILE(VSQO)    FROM (VSR) KEYFROM(VSQA.KEY);
        GOTO AGAIN;
 /*------WRITE ERROR MESSAGE TO LOGFILE ---------------*/
ERROR3: ERROR = MAX(ERROR,3); /* BUFFER RECORD TOO SHORT    */
ERROR2: ERROR = MAX(ERROR,2); /* FEWER KEYS THAN RECORDS    */
ERROR1: ERROR = MAX(ERROR,1); /* EYECATCHER HAS SLIPPED !!! */
        CARD = 'ERROR CODE:-' || ERROR;
        IF ERROR > 0 THEN WRITE FILE (LOGFILE) FROM (CARD);
 /*------------ FINISH --------------------------------*/
STOP:   END;
```

A.3.5 DBTMOV1 PROGRAM SOURCE

```
DBTMOV1: PROC OPTIONS(MAIN REENTRANT);
 /*******************************************************/
 /*                                                     */
 /* DBTMOV1:                        17 AUGUST 1983      */
 /*     - PLI CICS COMMAND LEVEL PROGRAM                */
 /*     - GET AUTOMATICALLY INITIATED EVERY             */
 /*       5 MINUTES                                     */
 /*     - SUBROUTINES LINKED VIA CICS: DBTOPN1          */
 /*                                    DBTCLS1          */
 /*     - INPUT KEY SEQUENCED 6K VSAM RECORDS OF        */
 /*       DBTVSQ0, KEY IN FIRST 39 BYTES                */
 /*     - OUTPUT KEY SEQUENCED 6K VSAM RECORDS TO       */
 /*       DBTVSQ1, KEY IN FIRST 39 BYTES                */
 /*     - NO DATA TRANSFORMATIONS ARE DONE              */
 /*     - COPIED RECORDS ARE DELETED FROM DBTVSQ0       */
 /*     - THE PROGRAM CHECKS IF DBTVSQ0 IS OPEN         */
 /*       IF SO IT ASSUMES THAT DBTBAT1 IS RUNNING      */
 /*       STOPS TO RETRY IN 5 MINUTES TIME              */
 /*     - FOR EVERY DOCUMENT (CHANGE IN FIRST 35 BYTES  */
 /*       OF KEY) IT INITIATES CICS TRANSACTION DBTS    */
 /*       (PROGRAM DBTMST1) WITH THE KEY OF THE FIRST   */
 /*       RECORD OF THE DOCUMENT AS THE START KEY       */
 /*     - CLOSES DBTVSQ0                                */
 /*     - STARTS ITSELF IN 5 MINUTES                    */
 /*                                                     */
 /*******************************************************/
DCL (LENGTH,STG,CSTG,ADDR,MAX)   BUILTIN;
%INCLUDE DBTVSQ;                     /* DBTVSQ0 AND DBTVSQ1   */
/****************************************************/
/*                                                  */
/*    DBTOC:  CONTROL BLOCK                         */
/*            USED FOR REMEMBERING KEY, HEADER      */
/*            AND POINTERS TO OTHER CONTROL BLOCKS  */
/****************************************************/
 DCL Q0Q1PTR POINTER;
 DCL
   1 Q0Q1    BASED(Q0Q1PTR),
   2 KEY,               /* KEY 39 CHARACTERS                      */
     3 DATE CHAR(5),        /* DATE                               */
     3 TIME CHAR(9),        /* TIME                               */
     3 INTTYPE CHAR(1),     /* INTERFACE TYPE                     */
     3 OSN CHAR(8),         /* OSN NAME                           */
     3 USER CHAR(8),        /* USER NAME FOR DISTRIBUTION         */
     3 INTYPE BIT(16),      /* INPUT DOCTYPE DIA-CODED            */
     3 OUTYPE BIT(16),      /* OUPUT DOCTYPE DIA-CODED            */
     3 SEQNO FIXED BIN(15), /* SEQUENCE NUMBER X'0000' TO X'FFFF' */
     3 CHFLAGF CHAR(1),     /* CHAINING FLAG 1                    */
     3 CHFLAGL CHAR(1),     /* CHAINING FLAG 2                    */

   2 HEAD,                  /* ONLY FOR FIC OR LIC                */

      3 RECTYP    CHAR(1),  /* RECORD TYPE A ->                   */
      3 PROFLAG   CHAR(1),  /* RECORD SIZE IN CARD IMAGE UNITS    */
      3 PAGEL CHAR(3),      /* DECIMAL PAGELENGTH IN LINES/PAGE   */
      3 PAGEW CHAR(3),      /* DECIMAL PAGEWIDTH IN CHAR/LINE     */
      3 DISNAM CHAR(8),     /* UNUSED                             */
      3 EYECAT    CHAR(6),  /* 'HEADER' CONSTANT                  */
      3 DOCNAM CHAR(15),    /* DOCUMENT NAME                         */
      3 RESER CHAR(43);     /* RESERVED FILED LENGHT                 */

  DCL VIPTR POINTER,        /* POINTER TO INPUT  AREA                */
      VOPTR POINTER,        /* POINTER TO OUTPUT AREA                */
      OPENPTR POINTER,      /* POINTER TO OPENBLK COMMAREA           */

 1 Q0Q1B BASED(Q0Q1PTR),    /* OVERLAY                               */

   2 KEY,                   /* KEY 39 CHARACTERS                     */
     3 KEY1 CHAR(35),       /* FIXED PART FOR ONE DOC                */
     3 KEY2 CHAR(4),        /* USER NAME FOR DISTRIBUTION            */

   2 HEAD CHAR(80),         /* ONLY FOR FIC OR LIC                   */

 1 Q0Q1C BASED(Q0Q1PTR),    /*FIC OR OIC OR LIC                      */

   2 KEY CHAR(39);          /* KEY 39 CHARACTERS                     */

/*****************************************************/
/*    OPENBLK SUBROUTINE COMMUNICATION FIELD        */
/*            FOR OPEN AND CLOSE MACRO.             */
/*            LAYOUT CORRESPONDS WITH DATA          */
/*            DEFINITION FOR DFHOC MACRO            */
/*****************************************************/
 DCL
   1 OPENBLK BASED(OPENPTR),

     2 DBNAME CHAR(8),      /* DATA BASE NAME TO BE OPENED/CLOSED  */
     2 RC     BIT(8),       /* RETURN CODE FIELD X'00' OKAY        */
     2 FCT    BIT(24),      /* FCT ENTRY ON RETURN                 */
     2 FFF    BIT(24);      /* X'FFFFFF' MARKS END OF CONTROL BLK  */

/*****************************************************/
/*    GET STORAGE AREAS                              */
/*****************************************************/
  EXEC CICS GETMAIN SET(Q0Q1PTR) LENGTH(125) INITIMG('00000000'B);
  EXEC CICS GETMAIN SET(OPENPTR) LENGTH(15) INITIMG('00000000'B);
  EXEC CICS GETMAIN SET(VIPTR) LENGTH(6002) INITIMG('00000000'B);
/*****************************************************/
/*-------------- OPEN MACRO ATTEMPT ----------------*/
/*****************************************************/
  DBNAME = 'DBTVSQ0 ';
  FFF = (3)'11111111'B;
   EXEC CICS LINK PROGRAM('DBTOPN1') COMMAREA(OPENBLK) LENGTH(15);
   IF RC ¬= '00000000'B THEN GOTO CLOSE;
/*****************************************************/
/* START READING DATASET AT BEGINNING               */
/*****************************************************/
  EXEC CICS STARTBR DATASET('DBTVSQ0') KEYLENGTH(0) RIDFLD(VIKEY)
  GENERIC;

NEXTREC: EXEC CICS HANDLE CONDITION ENDFILE(STOP);
   EXEC CICS READNEXT DATASET('DBTVSQ0') INTO(VI)
   RIDFLD(VIKEY);
   IF VIDKEY = ((39)'11111111'B) THEN GOTO STOP; /* DUMMY RECORD HIT*/
   IF VICKEY ¬= Q0Q1B.KEY.KEY1                   /* DOC# CHANGE     */
   & VI3KEY.CHFLAGF ¬= '1'                       /* INPUT NOT FIC   */
   THEN DO;
      /* NO FIC ON NEW DOCUMENT ERROR CODE TO BE ADDED */
   END;
   IF VI3KEY.CHFLAGF = '1'                /* NEW DOCUMENT WITH FIC  */
   & Q0Q1.CHFLAGL ¬= '1'                  /* SHOULD HAVE HAD LIC    */
   & Q0Q1C.KEY ¬= ''                      /* ONLY ON FIRST DOCUMENT */
   THEN DO;
      /* NO LIC ON LAST DOCUMENT ERROR CODE TO BE ADDED */
   END;
      /* SAVE KEY FOR CHECK OF FIC/LIC                  */
   Q0Q1C.KEY = VIKEY;
      /* SAVE HEADER FOR LOST LIC RECOVERY              */
   IF VI3KEY.CHFLAGF= '1' THEN Q0Q1B.HEAD = VICHEAD;
/*--------- COPY RECORDS TO DBTVSQ1  ----------*/
   EXEC CICS WRITE DATASET('DBTVSQ1') FROM(VI)
   RIDFLD(VIKEY) LENGTH(39+2+LENGTH(VIDATA));
/****************************************************/
/* AT END OF DOCUMENT START DBTTRN1                 */
/*                AND DELETE RECORDS ON DBTVSQ0     */
/****************************************************/
   IF VI3KEY.CHFLAGL='1'                   /*  LIC ?                */
   THEN DO;
      IF VI3KEY.CHFLAGF ¬= '1'             /* REBUILD FIRST KEY     */
      THEN DO;
        VI3KEY.SEQNO = 0;
        VI3KEY.CHFLAGF = '1';
      END;
/*--------- MASS DELETE DBTVSQ0 DOCUMENT-------*/
 EXEC CICS DELETE DATASET('DBTVSQ0') RIDFLD(VICKEY)
           KEYLENGTH(35) GENERIC;
/*--------- PASS KEY TO DBTTRN1 ---------------*/
    EXEC CICS START TRANSID('DBTS')  FROM(VIKEY) LENGTH(39);
/*  TERMID('L430') */                  /* ONLY FOR TESTING           */
    VI3KEY.SEQNO = Q0Q1.KEY.SEQNO;     /* TO PREVENT ILLOGIC ON READ */
    VI3KEY.CHFLAGF = Q0Q1.KEY.CHFLAGF; /* TO PREVENT ILLOGIC ON READ */
    END;
    GOTO NEXTREC;
STOP:
 EXEC CICS ENDBR DATASET('DBTVSQ0');
/*--------- CLOSE DATASET ---------------------*/
CLOSE:
   EXEC CICS LINK PROGRAM('DBTCLS1') COMMAREA(OPENBLK) LENGTH(15);
   IF RC ¬= '00000000'B THEN GOTO ERR3;
   GOTO LAST;
ERR3: GOTO LAST;          /* OPEN/CLOSE ERROR  */
/*----- START ITSELF IN 5 MINUTES AND STOP ----*/
LAST:
/*                                                       */
/*    RESTART OF DBTM COMMENTED OUT FOR TESTING ...WBW   */
/*                                                       */
/*    EXEC CICS START TRANSID('DBTM') INTERVAL(000500);  */
/*                                                       */
/*                                                       */
/*                                                       */
      END;
```

A.3.6 DBTOPN1 PROGRAM SOURCE

```
OPN  TITLE '*DBTOPN1* - PERFORM CICS OPEN FOR DATASET*'
         PRINT NOGEN
         GBLB  &DFHEIMX     INDICATE MIXED MODE
&DFHEIMX SETB  1            INDICATE MIXED MODE
         COPY  DFHCSADS     COPY CSA DEFINITION
         COPY  DFHTCADS     COPY TCA DEFINITION
         SPACE
DBTOPN1  CSECT
***********************************************************************
* DBTOPN1:                                                            *
*   ASSEMBLER MIXED MODE TRANSACTION                                  *
*   SUBROUTINE TO DO A CICS OPEN                                      *
*                                                                     *
*   INPUT PARMLIST:                                                   *
*                    -> OPENBLK CONTROL BLOK POINTER CONTAINING:      *
*                    -> DATSETNAME (8 BYTES)                          *
*                    -> RC FIELD   (1 BYTE)                           *
*                    -> FCT FIELD  (3 BYTES)                          *
*                    -> FFF FIELD  (3 BYTES) TO INIDCATE END OF BLOCK *
*                                                                     *
*   OUTPUT: RC       HAS RETURN CODE.                                 *
*             = 8 : BAD RETURN FROM OPEN MACRO                        *
*             = 0 : OKAY                                              *
*                                                                     *
***********************************************************************
         SPACE
         PRINT ON
         L     8,DFHEICAP        OPENBLK ADDRESS
         SPACE
         DFHOC TYPE=OPEN,          ONLY OPEN                           X
               DATASET=DATABASE,    THIS IS FOR A VSAM DATASET         X
               LISTADR=8          POINTER TO PARM LIST
         EXEC CICS RETURN
         EJECT
         PRINT ON
         END
```

A.3.7 DBTCLS1 PROGRAM SOURCE

```
CLO  TITLE '*DBTCLS1* - PERFORM CICS CLOSE FOR DATABASE*'
         PRINT NOGEN
         GBLB  &DFHEIMX     INDICATE MIXED MODE
&DFHEIMX SETB  1            INDICATE MIXED MODE
         COPY  DFHCSADS     COPY CSA DEFINITION
         COPY  DFHTCADS     COPY TCA DEFINITION
DBTCLS1  CSECT
************************************************************************
* DBTCLS1                                                              *
*   ASSEMBLER SUBROUTINE, MIXED MODE TRANSACTION                       *
*                                                                      *
*   SUBROUTINE TO DO A CICS CLOSE                                      *
*                                                                      *
*   INPUT PARMLIST:                                                    *
*                    -> OPENBLK CONTROL BLOK POINTER CONTAINING:       *
*                    -> DATSETNAME (8 BYTES)                           *
*                    -> RC FIELD   (1 BYTE)                            *
*                    -> FCT FIELD  (3 BYTES)                           *
*                    -> FFF FIELD  (3 BYTES) INDICATING END OF BLOCK   *
*                                                                      *
*   OUTPUT: RC       HAS RETURN CODE.                                  *
*             = 8 : BAD RETURN FROM OPEN MACRO                         *
*             = 0 : OKAY                                               *
*                                                                      *
************************************************************************
         SPACE
         PRINT ON
         L     8,DFHEICAP       OPENBLK ADDRESS
         DFHOC TYPE=CLOSE,        ONLY CLOSE                           X
               DATASET=DATABASE,   THIS IS FOR A VSAM DATASET          X
               LISTADR=8         POINTER TO PARM LIST
         EXEC CICS RETURN
         EJECT
         PRINT ON
         END
```

A.3.8 DBTMST1 PROGRAM SOURCE

```
DBTMST1: PROC OPTIONS(MAIN);
 /*********************************************************/
 /*                                                       */
 /* DBTMST1: DBTMST1 PROGRAM SOURCE FOR TRANSACTION DBTS  */
 /*                                                       */
 /*    INPUT:  39-BYTE FIC KEY START DATA                 */
 /*                                                       */
 /*    OUTPUT: 39-BYTE FIC KEY START DATA                 */
 /*                                                       */
 /*    FUNCTION: 1) RETRIEVE VSQ1 REQUEST FIC KEY         */
 /*              2) OBTAIN FIC RECORD                     */
 /*              3) DETERMINE CORRECT TRANSFORM PROGRAM   */
 /*                    FROM KEY AND PROFILE PARAMETERS    */
 /*              4) TRANSFER CONTROL TO CORRECT           */
 /*                    TRANSFORM, PASSING FIC KEY         */
 /*                                                       */
 /*********************************************************/
%INCLUDE DBTVSQ;  /* --- DBTVSQ0/DBTVSQ1 I/O OVERLAYS --- */
 DCL (LENGTH,STG,CSTG,ADDR) BUILTIN;
 DCL KEYLEN FIXED BIN(15) INIT(39);
 DCL VOPTR POINTER,
     VIPTR POINTER;
 /* --- ALLOCATE INPUT RECORD STORAGE --- */
 EXEC CICS GETMAIN SET(VIPTR) LENGTH(6002) INITIMG('00000000'B);
 /* --- RETRIEVE FIC KEY --- */
 EXEC CICS RETRIEVE INTO(VIG) LENGTH(KEYLEN);
 /* --- OBTAIN FIC RECORD --- */
 EXEC CICS STARTBR DATASET('DBTVSQ1') RIDFLD(VICKEY)
      KEYLENGTH(35) GENERIC EQUAL;
 EXEC CICS READNEXT DATASET('DBTVSQ1') INTO(VIG)
           RIDFLD(VICKEY);
 /* --- ENDBROWSE TO PERMIT VALID TRANSFORM STARTBROWSE --- */
 EXEC CICS ENDBR DATASET('DBTVSQ1');
 /* ------- PASS KEY TO TRANSFORM ------- */
 /* TRANSFORM SELECTION DEPENDENT ON:     */
 /*          VID.INTYPE                   */
 /*          VID.OUTYPE                   */
 /*          VID.RECTYP                   */
 /* 'IF THEN GOTO' STRUCTURE INSTEAD OF   */
 /* 'IF THEN DO  ' TO AVOID COMPILER      */
 /* SYNTAX ERRORS DUE TO TRANSLATOR       */
 /* COMMENT AND DO BLOCK                  */
 /* ------------------------------------- */
 /* --- TRANSFORM 1 : X'02', X'OC', 'A'   */
 /* 1403 PRINT DATA ===> 1403 DCA L2      */
 IF ((VID.INTYPE='0000000000001100'B) &
            (VID.OUTYPE='0000000000000010'B) &
            (VID.RECTYP='A')) THEN GOTO TRN1;
 /* --- DEFAULT TRANSFORM : (TRANSFORM 1) */
      EXEC CICS XCTL PROGRAM('DBTTRN1') COMMAREA(VIKEY) LENGTH(39);
 /* --- TRANSFORM TRANSFER LIST --- */
TRN1: EXEC CICS XCTL PROGRAM('DBTTRN1') COMMAREA(VIKEY) LENGTH(39);
TRN2:
TRN3:
TRN4: /* ......... ETC. */
  END;
```

A.3.9 DBTTRN1 PROGRAM SOURCE

The DCA initial multibyte controls are constant, independent of the input profile information. A subset of the 1403 and ANSI design conversions are made.

```
DBTTRN1: PROC(STPOINT) OPTIONS(MAIN);
/********************************************************************/
/*                                                                  */
/* DBTTRN1: PL1 PROGRAM SOURCE DBTTRN1 FOR TRANSACTION DBTS         */
/*                                                                  */
/* INPUT:   6000 BYTE KEY SEQUENCED VSAM RECORDS FROM VSQ1          */
/*    1) KEYS WITH USER AND PROCESSING DATA                         */
/*    2) PROFILE DATA ON FIRST AND LAST RECORD                      */
/*    3) IRS SEPARATED PRINTLINES                                   */
/*                                                                  */
/* OUTPUT:  4088 BYTE KEY SEQUENCED VSAM RECORDS TO VSQ1            */
/*    1) KEYS WITH USER AND PROCESSING DATA                         */
/*    2) PROFILE DATA ON FIRST AND LAST RECORD                      */
/*    3) DCA LEVEL 2 DOCUMENT INCLUDING PRESET INITIAL DATA         */
/*                                                                  */
/* PROCEDURE:                                                       */
/* 1) THE VSQ1 BLOCKED PRINT LINE MAILBOX REQUEST IS IDENTIFIED     */
/*    BY THE START DATA PASSED FROM DBTMST1                         */
/* 2) THE DOCUMENT IS CONVERTED TO A DCA LEVEL 2 DATASTREAM         */
/*    AND IS WRITTEN TO VSQ1 IN UNITS OF A CONVENIENT SIZE          */
/* 3) THE INPUT RECORDS ARE DELETED FROM VSQ1                       */
/* 4) PROGRAM DBTSND1 IS INITIATED WITH FIRST KEY START DATA        */
/*                                                                  */
/********************************************************************/
/*-------- CONTROL BLOCK -------------------------------------------*/
DCL XFPTR POINTER;
DCL 1 XF BASED(XFPTR),
     3 DOCCHAR CHAR(1),
     3 CHFLAGF CHAR(1),
     3 SEQNO FIXED BIN(15),
     3 DOC1PTR POINTER,
     3 DOC2PTR POINTER,
     3 VIPTR   POINTER,
     3 VOPTR   POINTER,
     3 DOCPTR1 FIXED BIN(31),
     3 DOCPTR2 FIXED BIN(31),
    1 DOCBIT BIT(8) BASED(XFPTR);
DCL PROREM CHAR(80);
DCL XFLEN FIXED BIN(15);
DCL KEYLEN FIXED BIN(15) FIXED INIT(39);
DCL VICREM CHAR(35);
DCL TDATAREM CHAR(39);
DCL SDATAREM CHAR(35);
/*-------- WORK AREAS --- BASED FOR DEBUGGING ----------------------*/
DCL 1 DOCWORK1 BASED(DOC1PTR) CHAR(6000) VARYING; /*INPUT WORK AREA*/
DCL 1 DOCWORK2 BASED(DOC2PTR) CHAR(4090) VARYING; /*OUTPUT WORK AREA*/
/*-------- CONTROL CHARACTER OVERLAYS ------------------------------*/
DCL 1 BIT,
     2 NL  BIT(8) INIT('00010101'B),
     2 FF  BIT(8) INIT('00001100'B),
     2 CR  BIT(8) INIT('00001101'B),
    1 CHAR BASED(ADDR(BIT)),
     2 NL CHAR(1),
     2 FF CHAR(1),
     2 CR CHAR(1),
/*-------- DCA INITIAL DATA OVERLAYS -------------------------------*/
    1 DCAINI,
      3 SEA1 BIT(48)
   INIT('001010111101001000000100100001010000000000000000'B),
      3 SHM1 BIT(32)  INIT('00101011110100100000011000010001'B),
      3 SHM2 BIT(32)  INIT('00000000000000010010111111010000'B),
      3 SVM1 BIT(32)  INIT('00101011110100100000011001001001'B),
      3 SVM2 BIT(32)  INIT('00000010110100000011110111100000'B),
      3 SPPS1 BIT(32) INIT('00101011110100100000011001000000'B),
      3 SPPS2 BIT(32) INIT('00101111110100000011110111100000'B),
      3 SCG1 BIT(32)  INIT('00101011110100010000011000000001'B),
      3 SCG2 BIT(32)  INIT('00000000110101110000000100001000'B),
  1 DCACHAR BASED(ADDR(DCAINI)),
      3 SEA CHAR(6),       /* X'2BD204850000'     */
      3 SHM CHAR(8),       /* X'2BD2061100012FD0' */
      3 SVM CHAR(8),       /* X'2BD2064902D03DE0' */
      3 SPPS CHAR(8),      /* X'2BD206402FD03DE0' */
      3 SCG CHAR(8);       /* X'2BD1060100D70108' 1403 PRINT */
/*-------- VSQ0/VSQ1 I/O RECORD OVERLAYS --------------------------*/
%INCLUDE DBTVSQ;
DCL (SUBSTR,LENGTH,ADDR,CSTG             ) BUILTIN;
/*-------- START KEY VARIABLE -------------------------------------*/
DCL STPOINT POINTER,
    STARTKEY CHAR(39) BASED(STPOINT);
/*-------- ALLOCATE STORAGE AREAS ----------------------------------*/
  XFLEN = CSTG(XF);
  EXEC CICS GETMAIN SET(XFPTR)  LENGTH(XFLEN) INITIMG('00000000'B);
  EXEC CICS GETMAIN SET(DOC1PTR) LENGTH(6002) INITIMG('00000000'B);
  EXEC CICS GETMAIN SET(DOC2PTR) LENGTH(4090) INITIMG('00000000'B);
  EXEC CICS GETMAIN SET(VIPTR)   LENGTH(6002) INITIMG('00000000'B);
  EXEC CICS GETMAIN SET(VOPTR)   LENGTH(6002) INITIMG('00000000'B);
  VOD.CHFLAGF = '1';
  VOD.CHFLAGL = '0';
    /* THIS IS FOR BUG IN PL1 ONLY, NO CONCATENATE TO NULL STRING */
  DOCWORK2 = 'Z';
/*-------- RETRIEVE START DATA (IF PRIMARY   TRANSFORM) -----------*/
/*XXXX CICS RETRIEVE INTO(VIG) LENGTH(KEYLEN)*/
/*-------- RETRIEVE START DATA (IF SECONDARY TRANSFORM) -----------*/
  VICKEY = STARTKEY;
  SDATAREM = SUBSTR(VI,1,35);
  EXEC CICS STARTBR DATASET('DBTVSQ1') RIDFLD(VICKEY)
  KEYLENGTH(35) GENERIC EQUAL;
GTDOC: /*--------- GET NEXT INPUT RECORD -------*/
/*****************************************************************/
/* CHAINFLAG LAST IS SET BEFORE THE READ, AND RESET AFTER THE READ,*/
/* SO THAT IF THERE IS NO MORE INPUT DATA, IT IS POSSIBLE TO      */
/* BRANCH TO WRITE THE REMAINING OUTPUT DATA, WITHOUT FORGETING   */
/* THAT THIS IS THE LAST OUTPUT RECORD.                           */
/*****************************************************************/
  VOD.CHFLAGL = '1';
  EXEC CICS HANDLE CONDITION NOTFND(PTDOC);
  VICREM = VICKEY;
  EXEC CICS READNEXT DATASET('DBTVSQ1') INTO(VIG)
            RIDFLD(VICKEY);
  EXEC CICS HANDLE CONDITION NOTFND;
  IF VICREM ¬= VICKEY THEN GOTO PTDOC;
  IF VID.CHFLAGF = '1' THEN PROREM = SUBSTR(VIDATA,1,80);
  VOD.CHFLAGL = '0';
```

```
/*-------- SET INPUT WORK AREA --------------------------------------*/
  DOCPTR2 = 0;
  DOCWORK1 = VIDATA;
  IF VID.CHFLAGF = 1 | VID.CHFLAGL = 1 THEN
     DOCWORK1 = SUBSTR(VIDATA,81,LENGTH(VIDATA)-80);
  IF VID.CHFLAGF = '1' THEN
  DOCWORK2 = '1' || SEA || SHM || SVM || SPPS || SCG;
FNDPRT: /*--------- FIND NEXT PRINT LINE ---------*/
   IF DOCPTR2  >= LENGTH(DOCWORK1) THEN GOTO GTDOC;
    /* DOCPTR1 GIVES PREVIOUS FOUND IRS, DOCPTR2 GIVES LAST IRS */
    DOCPTR1 = DOCPTR2;
    DOCPTR2 = DOCPTR2 + 1;
    DO DOCPTR2 = DOCPTR2 TO LENGTH(DOCWORK1);
     DOCCHAR = SUBSTR(DOCWORK1,DOCPTR2,1);
     IF DOCBIT = '00011110'B THEN GOTO IRSFND;  /* IRS FOUND        */
    END;
  /*RECORD DID NOT END WITH IRS */
    DOCPTR2 = DOCPTR2 + 1;
IRSFND:
  IF (LENGTH(DOCWORK2) + DOCPTR2 - DOCPTR1 + 1) > 4008 THEN DO;
PTDOC: /*------ WRITE RECORD IF IT IS FULL ------------*/
   VOKEY = VICREM || SUBSTR(VOKEY,36,4); /* SET APPROXIMATE KEY */
   VODATA = SUBSTR(DOCWORK2,2,LENGTH(DOCWORK2)-1);
   VOD.INTYPE = '0000000000000010'; /* FF TEXT */
/*-------- ADD PROFILE DATA IF FIC/LIC/OIC -------------------------*/
   IF VOD.CHFLAGF = '1' | VOD.CHFLAGL = '1' THEN
           VODATA = PROREM || VODATA;
SKIPADD: EXEC CICS WRITE DATASET('DBTVSQ1') FROM(VOG) RIDFLD(VOKEY)
    LENGTH(39 + 2 + LENGTH(VODATA));
/*-------- REMEMBER TDQ2 DATA ---------------------------------------*/
    IF VOD.CHFLAGF = '1' THEN TDATAREM = SUBSTR(VO,1,35);
    IF VOD.CHFLAGL = '1' THEN GOTO ENDAT;
    VOD.SEQNO = VOD.SEQNO + 1;
    VOD.CHFLAGF = '0';
    /* THIS IS FOR BUG IN PL1 ONLY, NO CONCATENATE TO NULL STRING  */
    DOCWORK2 = '1';
  END; /*------- MODIFY PRINTLINE ACCORDING TO CARRIAGE CONTROL ---*/
/**********************************************************************/
/* DOCPTR1 POINTS TO PREVIOUS IRS                                     */
/* DOCPTR2 POINTS TO NEXT IRS                                         */
/**********************************************************************/
  DOCCHAR = SUBSTR(DOCWORK1,DOCPTR1 + 1,1);
      IF DOCPTR2  >= LENGTH(DOCWORK1)
      & VID.CHFLAGL = 1 THEN DO;
      /* DO NOT INSERT DCA CONTROLS AT END OF DOCUMENT */
       SELECT(DOCBIT);    /* SP1  ->  NL  */
        WHEN ('00001011'B)
        DOCWORK2 = DOCWORK2 || CHAR.NL ||
        SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2);
        WHEN ('00010011'B) DO;
         DOCWORK2 = DOCWORK2 || CHAR.NL || CHAR.NL ||
         SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2);
        END;
        WHEN ('00011011'B) DO;
         DOCWORK2 = DOCWORK2 || CHAR.NL || CHAR.NL || CHAR.NL ||
         SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2);
        END;
        WHEN ('10001011'B)
        DOCWORK2 = DOCWORK2 || CHAR.FF ||
        SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2);
        WHEN ('00000001'B) DO;
         EXEC CICS ENTER TRACEID(2);
         DOCWORK2 = DOCWORK2 || CHAR.CR ||
         SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2);
        END;
        WHEN ('00001001'B) DO;
         DOCWORK2 = DOCWORK2 || CHAR.CR ||
         SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2);
        END;
        WHEN ('00010001'B) DO;
         DOCWORK2 = DOCWORK2 || CHAR.CR ||
         SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2);
        END;
        WHEN ('00000011'B) GOTO FNDPRT;    /*         -> DELETE  */
        OTHERWISE DOCWORK2 = DOCWORK2 || CHAR.NL ||
        SUBSTR(DOCWORK1,DOCPTR1+2,DOCPTR2-DOCPTR1-2);
       END;
      DOCWORK2 = DOCWORK2 || CHAR.CR || CHAR.FF;
      /* AT END OF DOCUMENT ADD CR AND FF */
      GOTO FNDPRT;
      END;
   SELECT(DOCBIT);    /* SP1  ->  NL  */
    WHEN ('00001011'B)
    DOCWORK2 = DOCWORK2 || CHAR.NL ||
    SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2);
    WHEN ('00010011'B) DO;
     DOCWORK2 = DOCWORK2 || CHAR.NL || CHAR.NL ||
     SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2);
    END;
    WHEN ('00011011'B) DO;
     DOCWORK2 = DOCWORK2 || CHAR.NL || CHAR.NL
     || CHAR.NL || SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2);
    END;
    WHEN ('10001011'B)
    DOCWORK2 = DOCWORK2 || CHAR.FF ||
    SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2);
    WHEN ('00000001'B) DO;
     DOCWORK2 = DOCWORK2 || CHAR.CR ||
     SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2);
    END;
    WHEN ('00001001'B) DO;
     DOCWORK2 = DOCWORK2 || CHAR.CR ||
     SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2) || CHAR.NL;
    END;
    WHEN ('00010001'B) DO;
     DOCWORK2 = DOCWORK2 || CHAR.CR ||
     SUBSTR(DOCWORK1,DOCPTR1 + 2,DOCPTR2 - DOCPTR1 - 2) ||
     CHAR.NL||CHAR.NL;
    END;
    WHEN ('00000011'B) GOTO FNDPRT;    /*         -> DELETE  */
    OTHERWISE DOCWORK2 = DOCWORK2 || CHAR.NL ||
    SUBSTR(DOCWORK1,DOCPTR1+2,DOCPTR2-DOCPTR1-2);
   END;
   GOTO FNDPRT;
ENDAT: /* ------- DELETE INPUT RECORDS ----------------*/
  EXEC CICS ENDBR DATASET('DBTVSQ1');
  /* ADD CODING FOR INCOMPLETE DOCUMENTS, NO FIC, NO LIC */
  EXEC CICS DELETE DATASET('DBTVSQ1')
      RIDFLD(SDATAREM)
      KEYLENGTH(35) GENERIC;
```

```
/*-------- TRANSFER CONTROL TO DBTSND1 ----------------------------*/
  EXEC CICS XCTL PROGRAM('DBTSND1') COMMAREA(TDATAREM) LENGTH(39);
  END;
```

A.3.10 DBTSND1 PROGRAM SOURCE

```
DBTSND1:    PROC(COMARPTR) OPTIONS(MAIN);
/****************************************************/
/*  DBTSND1:              17 AUGUST 1983            */
/*    - PLI CICS COMMAND LEVEL TRANSACTION          */
/*    - SUBROUTINES CALLED: APIACTIV                */
/*                          APIPURGE                */
/*                          APIDIS2                 */
/*                          APIFIL2                 */
/*                          APIPTDOC                */
/*                          APISUFIX                */
/*                          APILAST                 */
/*      ALL LINKED INTO MAIN ROUTINE                */
/*    - XCTL FROM DBTTRN1 WITH COMMAREA KEY         */
/*    - INPUT:-KEY ITEM AS COMMAREA                 */
/*            -VSAM KEY SEQUENCED RECORDS WITH      */
/*             MAXIMUM RECORDSIZE OF 4088 + 80 + 39 */
/*    - OUTPUT: L2DCA DATA IS PASSED THROUGH API    */
/*              TOGETHER WITH DIA                   */
/*              'REQUEST_DISTRIBUTION' OR           */
/*              'FILE' COMMAND,                     */
/*              AND DOCUMENT PROFILE.               */
/*    - RESPONSE TRANSACTION TRIGGERED BY DISOSS    */
/*      IS DBTR.                                    */
/*                                                  */
/****************************************************/
DCL (STG,SUBSTR,ADDR,LENGTH,CSTG,UNSPEC) BUILTIN;

/****************************************************/
/* CONTROL BLOCKS USED BY API ROUTINES              */
/****************************************************/
%INCLUDE   APICOMP;              /* COMMON PLI VERSION */
DCL   CBLEN FIXED BIN(15);       /* APICOMP LENGTH     */
%INCLUDE   APIRETP2;             /* RETRIEVE AREA      */
%INCLUDE   APIDPRP2;             /* PROFILE AREA       */
/****************************************************/
/* INPUT/OUTPUT AREAS FOR DBTVSQ1                   */
/****************************************************/
 DCL VIPTR POINTER;
 DCL VOPTR POINTER;
%INCLUDE   DBTVSQ;

DCL COMARPTR POINTER;
DCL 1 COMKEY CHAR(39) BASED(COMARPTR),
    1 COMB BASED(COMARPTR),
      2 COR CHAR(16),
    1 COMC BASED(COMARPTR),
      2 KEY35 CHAR(35);

DCL 1 NEXTRANS CHAR(4) INIT('DBTR');

DCL   DOCPTR POINTER;             /* AREA FOR PASSING DCA-L2 DATA */
DCL 1 DOCUMENT CHAR(4088) VARYING BASED(DOCPTR),
    1 DOC2 BASED(DOCPTR),
      2 DOC2FILL CHAR(2),         /* LENGTH FIELD                 */
      2 DOC2L2   CHAR(4088);      /* MAXIMUM LENGTH 4096 - 8      */

DCL 1 SONPARMS,                   /* PARAM. FOR DIA SIGNON        */
```

```
     2 DOCTYPE BIT(16) INIT('0000000000000010'), /* NOT USED      */
     2 LNAME BIT(8) INIT('00000110'B),        /* LENGTH NAME      */
     2 LPASS BIT(8) INIT('00000001'B),        /* LENGTH PASSW.    */
     2 NAME,                                  /* USED BY ACTIV.   */
      3 NAME1 CHAR(6) INIT('DIST01'),         /* SIGNON ID        */
      3 NAME2 BIT(16) INIT((2)'00000000'B),   /*                  */
     2 PASS,                                  /* NOT USED         */
      3 PASS1 CHAR(1) INIT('1'),              /* PASSWORD         */
      3 PASS2 BIT(56) INIT((7)'00000000'B);   /*                  */


DCL   Z64 BIT(64) INIT((8)'00000000'B);       /* FOR ACTIVATE     */

DCL   DIUBUF CHAR(1096) BASED(COMDIUP);       /* FOR DIA/DCA      */
                                              /* CONSTRUCTS       */
DCL   APIACTIV ENTRY EXTERNAL OPTIONS(ASM INTER), /* ACTIVATE     */
      APIRTRVE ENTRY EXTERNAL OPTIONS(ASM INTER), /* RETRIEVE     */
      APIRECVE ENTRY EXTERNAL OPTIONS(ASM INTER), /* RECEIVE      */
      APIGTCMD ENTRY EXTERNAL OPTIONS(ASM INTER), /* GET COMMAND  */
      APIFIL2  ENTRY EXTERNAL OPTIONS(ASM INTER), /* DIA "FILE"   */
      APIPURGE ENTRY EXTERNAL OPTIONS(ASM INTER), /* PURGE ALL    */
      APIDIS2  ENTRY EXTERNAL OPTIONS(ASM INTER), /* DIA "DIST"   */
      APIPTDOC ENTRY EXTERNAL OPTIONS(ASM INTER), /* PUT DOC      */
      APISUFIX ENTRY EXTERNAL OPTIONS(ASM INTER), /* DIU SUFFIX   */
      APILAST  ENTRY EXTERNAL OPTIONS(ASM INTER); /* API "LAST"   */

    CBLEN = CSTG(APICOM);
    EXEC CICS GETMAIN SET(COMPTR) LENGTH(CBLEN) INITIMG('00000000'B);
    EXEC CICS GETMAIN SET(COMDPRP) LENGTH(4096) INITIMG('00000000'B);
    EXEC CICS GETMAIN SET(COMDIUP) LENGTH(1096) INITIMG('00000000'B);
    EXEC CICS GETMAIN SET(DOCPTR) LENGTH(4090) INITIMG('00000000'B);
    EXEC CICS GETMAIN SET(VIPTR) LENGTH(6004) INITIMG('00000000'B);
    COMTRFLG = COMTRYES;

/*****************************************************/
/* ACTIVATE API FOR THIS TRANSACTION                 */
/*****************************************************/
ATIIN:
    CALL APIACTIV(DFHEIBLK,APICOM,NAME,Z64);
    EXEC CICS ENTER TRACEID(91);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
DISDOC:
/* NOW LOOK IF MORE DOCUMENTS ARE PRESENT ON DBTVSQ1             */
  EXEC CICS STARTBR DATASET('DBTVSQ1') KEYLENGTH(35) RIDFLD(KEY35)
  GENERIC EQUAL;
/*--------- READ FIRST REC AND CHECK FIC ------*/
NEXTREC: EXEC CICS HANDLE CONDITION ENDFILE(STOP);
         EXEC CICS HANDLE CONDITION NOTFND(STOP);
   EXEC CICS READNEXT DATASET('DBTVSQ1') INTO(VI)
   RIDFLD(COMKEY);
   IF VIDKEY = ((39)'11111111'B) THEN GOTO NODOC; /* DUMMY RECORD */
   IF VI3KEY.CHFLAGF ¬= '1'
   THEN DO;
      EXEC CICS ENTER TRACEID(1);
      /* NO FIC ON NEW DOCUMENT ERROR CODE TO BE ADDED */
   END;

   IF VI3KEY.CHFLAGF = '1'
   THEN DO;
    /* NOW BUILD DIA COMMAND FOR DISTRIBUTE AND CORRECT PARAMETERS */
    /* OR FILE DEPENDING ON DISFIL FIELD IN HEADER                      */
 DISTRIB:
      DPRCOR = COMB.COR;                          /* CORRELATION    */
      DPRRID = VID.USER;                          /* RECEPIENT ADDR*/
      DPRDDN = VID.OSN;                           /* DEST. NODE     */
      DPRSYS = 'IBM DCA-L2';                      /* SYSTEM CODE    */
      IF VID.INTYPE = '0000000000000010'B THEN    /* DCA-L2 ?       */
      DPRDOT = 2;
      ELSE GOTO INTYP;                        /*INVALID INPUT TYPE */
      DPRPGC = '00000001010100010000000100000000'B; /* X'01510100' */
      DPRDON = VI3DATA.DOCNAM;                    /* DOC. NAME      */
      DPRDONL = '00001111'B;                      /* LENGTH         */
      DPRSUB = 'MAILBOX PROJECT     ';            /* SUBJECT        */
      DPRSUBL = '00010100'B; /* 20 */             /* LENGTH         */
      DPRAUT = 'MAILBOX   ';                      /* AUTHOR         */
      DPRAUTL = '00001010'B; /* 10 */             /* LENGTH         */
      /* ACCESS CODE USED FOR FILE, COMMON ACCESS */
      DPRACC = '0000';                            /* ACCESS CODE    */
      DPRACCL = '00000100'B; /* 4 */              /* LENGTH         */
      IF VI3DATA.DISFIL = 'F'
      THEN DO;
           CALL APIFIL2(DFHEIBLK,APICOM,APIDPR,DIUBUF);
      END;
      ELSE DO;
           CALL APIDIS2(DFHEIBLK,APICOM,APIDPR,DIUBUF);
      END;
    EXEC CICS ENTER TRACEID(92);
      IF COMRETCD ¬= 0 THEN GOTO ERROR;
    END;
   IF VI3KEY.CHFLAGF= '1'                          /* FOR FIC OR  */
   |  VI3KEY.CHFLAGL= '1' THEN DO;                 /* LIC ONLY    */
      DOCUMENT = SUBSTR(VIDATA,81,(LENGTH(VIDATA)-80)); /*  DATA  */
    END;
    ELSE DO;
      DOCUMENT = SUBSTR(VIDATA,1,LENGTH(VIDATA));  /* DCA-L2 DATA */
    END;
    CBLEN = LENGTH(DOCUMENT);
                   /* DOC2L2 IS DCA-L2 DATA WITHOUT LENGTH FIELD */
    CALL APIPTDOC(DFHEIBLK,APICOM,DOC2L2,CBLEN,APIDPR);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    IF VI3KEY.CHFLAGL = '1'                          /* LIC      */
    THEN GOTO SEGL;
    GOTO NEXTREC;
SEGL:
    EXEC CICS ENDBR DATASET('DBTVSQ1');
    CBLEN = 0;
                                /* BUILD DOC. UNIT, SEGMENT LAST */
    CALL APIPTDOC(DFHEIBLK,APICOM,DOC2L2,CBLEN,APIDPR);
    EXEC CICS ENTER TRACEID(93);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    EXEC CICS ENTER TRACEID(94);
                                /* BUILD DIU SUFFIX              */
    CALL APISUFIX(DFHEIBLK,APICOM);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    GOTO LAST;

LAST:
                                /* TELL API TO PROCESS           */
    CALL APILAST(DFHEIBLK,APICOM,NEXTRANS);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    RETURN;

STOP:
 EXEC CICS ENDBR DATASET('DBTVSQ1');
    EXEC CICS ENTER TRACEID(95);
/*NO LAST IN CHAIN FOUND ON DBTVSQ1 FOR A DOCUMENT */
    RETURN;

NOHEAD:
/*NO HEADER RECORD FOUND ON DBTVSQ1 FOR A DOCUMENT */
    EXEC CICS ENTER TRACEID(96);
    RETURN;

NODOC:
/*NO MORE DOCUMENTS FOR DISTRIBUTION STOP */
    EXEC CICS ENTER TRACEID(97);
    RETURN;

INTYP:
/*INVALID INPUT DOCUMENT TYPE             */
    RETURN;

Q3BUS:
/*ANOTHER DBTSND1 IS BUSY GET OUT QUICKLY */
    RETURN;

ERROR:
    /* PURGE DELETES EVERYTHING ON THE APIQUEUE FOR THIS USER */
    /* A DIFFERENT WAY OF HANDLING API ERRORS SHOULD BE USED  */
    EXEC CICS ENTER TRACEID(98);
    CALL APIPURGE(DFHEIBLK,APICOM,NAME);
  END;
```

A.3.11 DBTRSP1 PROGRAM SOURCE

```
DBTRSP1:    PROC OPTIONS(MAIN);
/****************************************************/
/*  DBTRSP1:              17 AUGUST 1983            */
/*    - PLI CICS COMMAND LEVEL TRANSACTION          */
/*    - SUBROUTINES CALLED: APIRTRVE                */
/*                          APIACTIV                */
/*                          APIGTCMD                */
/*                          APIPURGE                */
/*                          APILAST                 */
/*      ALL LINKED INTO MAIN ROUTINE                */
/*    - DISOSS API RESPONSE TRANSACTION             */
/*    - INPUT -VSAM KEY SEQUENCED RECORDS WITH      */
/*             MAXIMUM RECORDSIZE OF 4088 + 80 + 39 */
/*            -RETRIEVE DATA OF DISOSS              */
/*    - OUTPUT: NONE                                */
/*                                                  */
/****************************************************/
DCL (STG,SUBSTR,ADDR,LENGTH,CSTG,UNSPEC) BUILTIN;

/****************************************************/
/* CONTROL BLOCKS USED BY API ROUTINES              */
/****************************************************/
%INCLUDE   APICOMP;              /* COMMON PLI VERSION */
DCL   CBLEN FIXED BIN(15);       /* APICOMP LENGTH     */
%INCLUDE   APIRETP2;             /* RETRIEVE AREA      */
%INCLUDE   APIDPRP2;             /* PROFILE AREA       */
/****************************************************/
/* INPUT/OUTPUT AREA FOR DISVSQ1                    */
/****************************************************/
 DCL VIPTR POINTER;
 DCL VOPTR POINTER;
%INCLUDE   DBTVSQ;

DCL   DOCPTR POINTER;             /* AREA FOR PASSING DCA-L2 DATA */
DCL 1 DOCUMENT CHAR(4088) VARYING BASED(DOCPTR),
    1 DOC2 BASED(DOCPTR),
      2 DOC2FILL CHAR(2),         /* LENGTH FIELD                 */
      2 DOC2L2   CHAR(4088);      /* MAXIMUM LENGTH 4096 - 8      */

DCL 1 SONPARMS,                   /* PARAM. FOR DIA SIGNON       */
     2 DOCTYPE BIT(16) INIT('0000000000000010'), /* NOT USED    */
     2 LNAME BIT(8) INIT('00000110'B),        /* LENGTH NAME    */
     2 LPASS BIT(8) INIT('00000001'B),        /* LENGTH PASSW.  */
     2 NAME,                                  /* USED BY ACTIV. */
      3 NAME1 CHAR(6) INIT('DIST01'),         /* SIGNON ID      */
      3 NAME2 BIT(16) INIT((2)'00000000'B),   /*                */
     2 PASS,                                  /* NOT USED       */
      3 PASS1 CHAR(1) INIT('1'),              /* PASSWORD      */
      3 PASS2 BIT(56) INIT((7)'00000000'B);   /*               */


DCL   Z64 BIT(64) INIT((8)'00000000'B);       /* FOR ACTIVATE  */

DCL   DIUBUF CHAR(1096) BASED(COMDIUP);       /* FOR DIA/DCA   */
                                              /* CONSTRUCTS    */
DCL   APIRTRVE ENTRY EXTERNAL OPTIONS(ASM INTER),  /* RETRIEVE     */
      APIACTIV ENTRY EXTERNAL OPTIONS(ASM INTER), /* ACTIVATE      */
      APIGTCMD ENTRY EXTERNAL OPTIONS(ASM INTER), /* GET COMMAND   */
      APIPURGE ENTRY EXTERNAL OPTIONS(ASM INTER), /* PURGE ALL     */
      APILAST  ENTRY EXTERNAL OPTIONS(ASM INTER); /* API "LAST"    */

    CBLEN = CSTG(APICOM);
    EXEC CICS GETMAIN SET(COMPTR) LENGTH(CBLEN) INITIMG('00000000'B);
    EXEC CICS GETMAIN SET(COMDPRP) LENGTH(4096) INITIMG('00000000'B);
    EXEC CICS GETMAIN SET(COMDIUP) LENGTH(1096) INITIMG('00000000'B);
    EXEC CICS GETMAIN SET(DOCPTR) LENGTH(4090) INITIMG('00000000'B);
    EXEC CICS GETMAIN SET(VIPTR) LENGTH(6004) INITIMG('00000000'B);
    COMTRFLG = COMTRYES;

/****************************************************/
/* GET DISOSS RETRIEVE DATA                         */
/****************************************************/
    CALL APIRTRVE(DFHEIBLK,APICOM);
    IF COMDLEN = 0 THEN GOTO ERROR;


RESPONSE:
    RETPTR = COMDPTR;


RDISTR:
    CALL APIACTIV(DFHEIBLK,APICOM,RETNAME,RETTIME);  /* ACTIVATE API*/
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    CALL APIGTCMD(DFHEIBLK,APICOM,APIDPR);           /* RECEIVE DATA*/
    IF COMRETCD ¬= 0 THEN GOTO ERROR;                /* OKAY ?      */
    IF COMDCMD ¬= COMDACK THEN GOTO ERROR;           /* ACKNOWLEDGE?*/
    IF DPREXCOD ¬= (3)'00000000'B THEN GOTO ERROR;   /* EXCEPT. CODE*/

/* PREVIOUS DISTRIBUTION OKAY NOW CLEANUP LAST DISTRIBUTED DOC*/
CLNUP:
    EXEC CICS STARTBR DATASET('DBTVSQ1') RIDFLD(RETDIUID)
    KEYLENGTH(16) GENERIC EQUAL;
    VIKEY=RETDIUID;
    EXEC CICS READNEXT DATASET('DBTVSQ1') RIDFLD(VIKEY)
    INTO(VI);
    EXEC CICS ENDBR DATASET('DBTVSQ1');
    EXEC CICS DELETE DATASET('DBTVSQ1') RIDFLD(VIKEY)
    KEYLENGTH(31) GENERIC;
    RETURN;
ERROR:
    /* PURGE DELETES EVERYTHING ON THE APIQUEUE FOR THIS USER */
    /* A DIFFERENT WAY OF HANDLING API ERRORS SHOULD BE USED  */
    CALL APIPURGE(DFHEIBLK,APICOM,NAME);
  END;
```

A.3.12 DBTSON1 PROGRAM SOURCE

Experience has shown that if distribute or file requests are placed on the API queue before this signon program has completed, then not only will those requests cause DISOSS transaction DSV1 to fail, but any subsequent signons will also fail. The only obvious course of action under these circumstances is to submit the DISOSS installation job which recreates the API queue.

```
DBTSON1:   PROC OPTIONS(MAIN);

 /*****************************************************/
 /*                                                   */
 /* DBTSON1:                         17 AUGUST 1983   */
 /*     - PLI CICS COMMAND LEVEL PROGRAM              */
 /*     - SUBROUTINES CALLED: APIRTRVE                */
 /*                           APIACTIV                */
 /*                           APISGNON                */
 /*                           APILAST                 */
 /*                           APIPURGE                */
 /*                           APIRECVE                */
 /*                           APIGTCMD                */
 /*     - INPUT RETRIEVE DATA FROM DISOSS             */
 /*     - SIGNON  COMMAND TO DISOSS                   */
 /*                                                   */
 /*****************************************************/
%INCLUDE   APICOMP;
%INCLUDE   APIRETP;
%INCLUDE   APIDPRP;

DCL (ADDR,LENGTH,CSTG,UNSPEC) BUILTIN;

DCL   CBLEN FIXED BIN(15);

DCL 1 SONPARMS,
     2 DOCTYPE BIT(16) INIT('0000000000000010'),
     2 LNAME BIT(8) INIT('00000110'B),
     2 LPASS BIT(8) INIT('00000001'B),
     2 NAME,
      3 NAME1 CHAR(6) INIT('DIST01'),
      3 NAME2 BIT(16) INIT((2)'00000000'B),
     2 PASS,
      3 PASS1 CHAR(1) INIT('1'),
      3 PASS2 BIT(56) INIT((7)'00000000'B);

DCL   DIUBUF CHAR(4096) BASED(COMDIUP);

DCL   Z64 BIT(64) INIT((8)'00000000'B);

DCL   APISGNON ENTRY EXTERNAL OPTIONS(ASM INTER), /* SIGNON        */
      APIRTRVE ENTRY EXTERNAL OPTIONS(ASM INTER), /* RETRIEVE      */
      APIACTIV ENTRY EXTERNAL OPTIONS(ASM INTER), /* ACTIVATE      */
      APIRECVE ENTRY EXTERNAL OPTIONS(ASM INTER), /* RECEIVE       */
      APIGTCMD ENTRY EXTERNAL OPTIONS(ASM INTER), /* GET COMMAND   */
      APIPURGE ENTRY EXTERNAL OPTIONS(ASM INTER), /* PURGE ALL     */
      APILAST  ENTRY EXTERNAL OPTIONS(ASM INTER); /* API "LAST"    */

    CBLEN = CSTG(APICOM);
    EXEC CICS GETMAIN SET(COMPTR) LENGTH(CBLEN) INITIMG('00000000'B);
    EXEC CICS GETMAIN SET(COMDPRP) LENGTH(4096) INITIMG('00000000'B);
    EXEC CICS GETMAIN SET(COMDIUP) LENGTH(4096) INITIMG('00000000'B);
    COMTRFLG = COMTRYES;

    CALL APIRTRVE(DFHEIBLK,APICOM);
    IF COMDLEN ¬= 0 THEN GOTO RESPONSE;

SIGNON:
    CALL APIACTIV(DFHEIBLK,APICOM,NAME,Z64);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    CALL APISGNON(DFHEIBLK,APICOM,DIUBUF,SONPARMS);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    GOTO LAST;

LAST:
    CALL APILAST(DFHEIBLK,APICOM,EIBTRNID);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    RETURN;


RESPONSE:
    RETPTR = COMDPTR;
    SELECT(RETDIUID);
      WHEN('01') GOTO RSGNON;
      OTHERWISE GOTO ERROR;
    END;

RSGNON:
    CALL APIACTIV(DFHEIBLK,APICOM,RETNAME,RETTIME);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    CALL APIGTCMD(DFHEIBLK,APICOM,APIDPR);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    IF COMDCMD ¬= COMDSON THEN GOTO ERROR;
    EXEC CICS START TRANSID('DBTM');
    RETURN;

ERROR:
    CALL APIPURGE(DFHEIBLK,APICOM,NAME);

  END;
```

A.3.13 DBTCLN1 PROGRAM SOURCE

```
DBTCLN1: PROC OPTIONS(MAIN REENTRANT);
 /*******************************************************/
 /*                                                     */
 /* DBTCLN1: DELETES ALL RECORDS ON VSQ0 AND VSQ1       */
 /*          AND PUTS ONE DUMMY RECORD ON EACH.         */
 /*                                                     */
 /*******************************************************/
DCL (LENGTH,STG,CSTG,ADDR,MAX)   BUILTIN;
%INCLUDE DBTVSQ; /*VSQ0 AND VSQ1   */
%INCLUDE DBTOC;  /*OPENBLK FOR OPEN/CLOSE MACRO */
  EXEC CICS GETMAIN SET(OPENPTR) LENGTH(15) INITIMG('00000000'B);
  EXEC CICS GETMAIN SET(Q0Q1PTR) LENGTH(125) INITIMG('00000000'B);
  EXEC CICS GETMAIN SET(VIPTR) LENGTH(6002) INITIMG('00000000'B);
/*--------- MASS DELETE VSQ1 ENTRIES ----------*/
 EXEC CICS STARTBR DATASET('DBTVSQ1') RIDFLD(VICKEY) GENERIC
 KEYLENGTH(0);
 EXEC CICS HANDLE CONDITION ENDFILE(VSQ1EN);
VSQ1:
 EXEC CICS READNEXT DATASET('DBTVSQ1') RIDFLD(VICKEY) INTO(VI);
 IF VIDKEY = ((39)'11111111'B) THEN GOTO VSQ1END;
 EXEC CICS DELETE DATASET('DBTVSQ1') RIDFLD(VICKEY)
           KEYLENGTH(35) GENERIC;
 GOTO VSQ1;
VSQ1EN:
   VIDKEY = (39)'11111111'B;
   EXEC CICS WRITE DATASET('DBTVSQ1') FROM(VI)
   RIDFLD(VIKEY) LENGTH(39);
VSQ1END:
 EXEC CICS ENDBR DATASET('DBTVSQ1');
/*--------- MASS DELETE VSQ0 ENTRIES ----------*/
  DBNAME = 'DBTVSQ0 ';
  RC = '11111111'B;
  FCT = (3)'11111111'B;
  FFF = (3)'11111111'B;
   EXEC CICS LINK PROGRAM('DBTOPN1') COMMAREA(OPENBLK) LENGTH(15);
   IF RC ¬= '00000000'B THEN GOTO STOP;
 EXEC CICS STARTBR DATASET('DBTVSQ0') RIDFLD(VICKEY) GENERIC
 KEYLENGTH(0);
 EXEC CICS HANDLE CONDITION NOTFND(VSQ0EN);
VSQ0:
 EXEC CICS READNEXT DATASET('DBTVSQ0') RIDFLD(VICKEY) INTO(VI);
 IF VIDKEY = ((39)'11111111'B) THEN GOTO VSQ0END;
 EXEC CICS DELETE DATASET('DBTVSQ0') RIDFLD(VICKEY)
           KEYLENGTH(35) GENERIC;
 GOTO VSQ0;
VSQ0EN:
   VIDKEY = (39)'11111111'B;
   EXEC CICS WRITE DATASET('DBTVSQ0') FROM(VI)
   RIDFLD(VIKEY) LENGTH(39);
VSQ0END:
 EXEC CICS ENDBR DATASET('DBTVSQ0');
STOP:
   EXEC CICS LINK PROGRAM('DBTCLS1') COMMAREA(OPENBLK) LENGTH(15);
   IF RC ¬= '00000000'B THEN GOTO ERR3;
ERR3: GOTO LAST;          /* OPEN/CLOSE ERROR  */
LAST: END;
```

A.3.14 DBTSOF1 PROGRAM SOURCE

Since a DIA Sign_Off is rarely required, the DBTSOF1 program is unlikely to be needed, and is not used in our system.

```
DBTSOF1:   PROC OPTIONS(MAIN);

 /*******************************************************/
 /*                                                     */
 /* DDBSOF1:                        17 AUGUST 1983      */
 /*     - PLI CICS COMMAND LEVEL PROGRAM                */
 /*     - SUBROUTINES CALLED: APIRTRVE                  */
 /*                           APIACTIV                  */
 /*                           APISNOFF                  */
 /*                           APILAST                   */
 /*                           APIPURGE                  */
 /*                           APIRECVE                  */
 /*                           APIGTCMD                  */
 /*     - INPUT RETRIEVE DATA FROM DISOSS               */
 /*     - SIGNOFF COMMAND TO DISOSS                     */
 /*                                                     */
 /*******************************************************/
%INCLUDE   APICOMP;
%INCLUDE   APIRETP;
%INCLUDE   APIDPRP;

DCL (ADDR,LENGTH,CSTG,UNSPEC) BUILTIN;

DCL   CBLEN FIXED BIN(15);

DCL 1 SONPARMS,
     2 DOCTYPE BIT(16) INIT('0000000000000110!'),
     2 LNAME BIT(8) INIT('00000110'B),
     2 LPASS BIT(8) INIT('00000001'B),
     2 NAME,
      3 NAME1 CHAR(6) INIT('DIST01'),
      3 NAME2 BIT(16) INIT((2)'00000000'B),
     2 PASS,
      3 PASS1 CHAR(1) INIT('1'),
      3 PASS2 BIT(56) INIT((7)'00000000'B);

DCL   DIUBUF CHAR(4096) BASED(COMDIUP);

DCL   Z64 BIT(64) INIT((8)'00000000'B);

DCL   APISNOFF ENTRY EXTERNAL OPTIONS(ASM INTER), /* SIGNOFF       */
      APIRTRVE ENTRY EXTERNAL OPTIONS(ASM INTER), /* RETRIEVE      */
      APIACTIV ENTRY EXTERNAL OPTIONS(ASM INTER), /* ACTIVATE      */
      APIRECVE ENTRY EXTERNAL OPTIONS(ASM INTER), /* RECEIVE       */
      APIGTCMD ENTRY EXTERNAL OPTIONS(ASM INTER), /* GET COMMAND   */
      APIPURGE ENTRY EXTERNAL OPTIONS(ASM INTER), /* PURGE ALL     */
      APILAST  ENTRY EXTERNAL OPTIONS(ASM INTER); /* API "LAST"    */

    CBLEN = CSTG(APICOM);
    EXEC CICS GETMAIN SET(COMPTR) LENGTH(CBLEN) INITIMG('00000000'B);
    EXEC CICS GETMAIN SET(COMDPRP) LENGTH(4096) INITIMG('00000000'B);
    EXEC CICS GETMAIN SET(COMDIUP) LENGTH(4096) INITIMG('00000000'B);
    COMTRFLG = COMTRYES;
    CALL APIRTRVE(DFHEIBLK,APICOM);
    IF COMDLEN ¬= 0 THEN GOTO RESPONSE;

SIGOFF:
    CALL APIACTIV(DFHEIBLK,APICOM,NAME,Z64);
    EXEC CICS ENTER TRACEID(3);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    EXEC CICS ENTER TRACEID(4);
    CALL APISNOFF(DFHEIBLK,APICOM);
    EXEC CICS ENTER TRACEID(5);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    GOTO LAST;

LAST:
    CALL APILAST(DFHEIBLK,APICOM,EIBTRNID);
    EXEC CICS ENTER TRACEID(3);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    EXEC CICS ENTER TRACEID(4);
    RETURN;


RESPONSE:
    RETPTR = COMDPTR;
    SELECT(RETDIUID);
      WHEN('01') GOTO RSGOFF;
      OTHERWISE GOTO ERROR;
    END;

RSGOFF:
    CALL APIACTIV(DFHEIBLK,APICOM,RETNAME,RETTIME);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    CALL APIGTCMD(DFHEIBLK,APICOM,APIDPR);
    IF COMRETCD ¬= 0 THEN GOTO ERROR;
    RETURN;

ERROR:
    CALL APIPURGE(DFHEIBLK,APICOM,NAME);

  END;
```

A.3.15 DBTTRT01 TRANSLATE TABLE

This job generates a special translate table to be added to the DISOSS system. DISOSS will use this table to translate documents from Codepage 264 (1403 TN) to GCID 337-256 (the Multi-Lingual Codepage), for delivery to subsystems (including DISOSS/PS and Displaywriter) which do not support Codepage 264.

```
//WTCR7A  JOB (0-863201),WRIGHT,MSGLEVEL=(1,1),MSGCLASS=A,
//      CLASS=A,REGION=1024K,NOTIFY=WTCR7
/*ROUTE PRINT RALYDPD3.WTCR7
/*ROUTE XEQ RALVSMV8
//*******************************************************************
//*             DISOSS V3 INSTALLATION ASSIST                       *
//*            TRANSLATE/PRINT FIDELITY TABLE                       *
//*                                                                 *
//*  THIS JOB ASSEMBLES AND LINK-EDITS TRANSLATE/PRINT FIDELITY     *
//*    TABLE DBTTRT01                                               *
//*                                                                 *
//*  LIBRARY - DISOSS30.DSVLOAD                                     *
//*  MEMBER  - DBTTRT01                                             *
//*                                                                 *
//* NOTE: REFER TO COMMENTS IN SOURCE CODE FOR PRINT FIDELITY TABLE *
//*       REGARDING OUTSTANDING DISOSS PROBLEM AS AT JAN 24 1984    *
//*                                                                 *
//*******************************************************************
//XLATE    PROC
//EXASM    EXEC PGM=IFOX00,PARM='OBJ,NODECK'
//SYSLIB   DD DSN=SYS1.MACLIB,DISP=SHR
//         DD DSN=DISOSS30.ADSVMAC,DISP=SHR
//SYSPRINT DD SYSOUT=A
//SYSPUNCH DD DUMMY
//SYSUT1   DD UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT2   DD UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSUT3   DD UNIT=SYSDA,SPACE=(1700,(600,100))
//SYSGO    DD UNIT=SYSDA,SPACE=(80,(500,50)),DISP=(,PASS),DSN=&&OBJ
//SYSIN    DD DDNAME=DSVIN
//LKED EXEC PGM=IEWL,
//       PARM='NCAL,MAP,LET,XREF,SIZE=(500K,100K)',REGION=130K
//SYSPRINT DD SYSOUT=A
//SYSUDUMP DD SYSOUT=A
//SYSLIB   DD DSN=DISOSS30.DSVLOAD,DISP=SHR
//SYSLIN   DD DSN=&&OBJ,DISP=(OLD,DELETE)
//SYSLMOD  DD DSN=DISOSS30.DSVLOAD(&MBR),DISP=SHR
//SYSUT1   DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//         PEND
//EXECPROC EXEC XLATE,MBR=DBTTRT01
//DSVIN DD *
******************************************************************
*
* $MOD(DBTTRT01)  COMP(ST)  PROD(DISOSS):
*
*      DESCRIPTIVE NAME: TRANSLATION/PRINT FIDELITY TABLE
*
*      COPYRIGHT: 5665-290 COPYRIGHT IBM CORP 1983
*            LICENSED MATERIAL - PROGRAM PROPERTY OF IBM
*            REFER TO COPYRIGHT INSTRUCTIONS FORM
*            NUMBER G120-2083
*
```

```
*      STATUS: VERSION 3 RELEASE 1
*
*      FUNCTION: THIS MODULE IS THE TRANSLATE / PRINT FIDELITY
*                TABLE THAT ALLOWS TRANSLATION
*                FROM XXXXX-00264 (1403 TN PRINT TRAIN)
*                TO   00337-00256 (INTERNATIONAL MULTILINGUAL)
*
*      NOTES:
*         DEPENDENCIES: NONE
*         RESTRICTIONS: NONE
*         REGISTER CONVENTIONS: NONE
*
*      INPUT: N/A
*
*      OUTPUT: N/A
*
*      EXIT CONDITIONS: N/A
*
*      EXTERNAL REFERENCES:
*
*         ROUTINES: NONE
*
*         DATA AREAS:
*           REFERENCED: NONE
*           MODIFIED: NONE
*
*         CONTROL BLOCKS:
*           REFERENCED: NONE
*
*           MODIFIED: NONE
*
*         TABLES: N/A
*
*         MACROS:
*           INTERNAL: NONE
*           EXTERNAL: NONE
*
*   CHANGE ACTIVITY:
*   $L0=ST       HD03102  052682 846301: DR-G
*   $P1=MPF0956  HD03102  090282 846301: OUTGCID COMMENT WRONG
*   $D1=B@@ST900 HD03102  120682 846301: CHANGE TO VERSION 2 TABLE
*   $P2=MPS0022  HD03102  120682 846301: SAME AS $D1 (B@@ST900)
*   $P3=MPS0009  HD03102  121382 846301: CORRECT XREF PROBLEM
*   $P4=MPS0270  HD03102  021083 846301: CHANGE TO REGISTERED GGID
*******************************************************************
DBTTRT01 CSECT          IN DSVXIDX TBLID=DBTTRT01
         DC    AL2(00264)          INGPID=00264
         DC    AL2(00337,00256)    OUTGCID=00337-00256  /*@P1C @P4C*/
DSVS6TRN DS    0XL256        THE TRANSLATION TABLE
         DC    X'000102030405060708090A0B0C0D0E0F'  /* 00 - 0F */
         DC    X'101112131415161718191A1B1C1D1E1F'  /* 10 - 1F */
         DC    X'202122232425262728292A2B2C2D2E2F'  /* 20 - 2F */
         DC    X'303132333435363738393A3B3C3D3E3F'  /* 30 - 3F */
* MAP CENT            (4A) TO CENT            (B0)
* MAP VERTICAL BAR    (4F) TO VERTICAL BAR    (BB)
         DC    X'404142434445464748490B04B4C4D4EBB'  /*40 - 4F*//*@D1C*/
* MAP EXCLAMATION     (5A) TO EXCLAMATION     (4F)
* MAP LOGICAL NOT     (5F) TO LOGICAL NOT     (BA)
         DC    X'5051525354555657585954F5B5C5D5EBA'  /*50 - 5F*//*@D1C*/
         DC    X'606162636465666768696A6B6C6D6E6F'  /*60 - 6F*//*@D1C*/
         DC    X'707172737475767778797A7B7C7D7E7F'  /*70 - 7F*//*@D1C*/
* MAP LEFT BRACE       (8B) TO LEFT BRACE       (C0)
* MAP LESS OR EQUAL    (8C) TO LEFT CHEVRON     (8A)
* MAP LEFT PAREN       (8D) TO LEFT PAREN       (4D)
* MAP PLUS             (8E) TO PLUS             (4E)
         DC    X'808182838485868788898AC08A4D4E8F'  /*80 - 8F*//*@D1C*/
* MAP RIGHT BRACE      (9B) TO RIGHT BRACE      (D0)
* MAP HOLLOW SQUARE    (9C) TO LOZENGE          (9F)
* MAP RIGHT PAREN      (9D) TO RIGHT PAREN      (5D)
* MAP PLUS OR MINUS    (9E) TO PLUS OR MINUS    (8F)
* MAP FILLED SQUARE    (9F) TO LOZENGE          (9F)
         DC    X'909192939495969798999AD09F5D8F9F'  /*90 - 9F*//*@D1C*/
* MAP HORIZONTAL BAR  (A0) TO MINUS             (6D)
* MAP DEGREE           (A1) TO DEGREE           (90)
* MAP BOX CNR-LWR LEFT(AB) TO PERIOD            (4B)
* MAP BOX CNR-UPR LEFT(AC) TO PERIOD            (4B)
* MAP LEFT BRACKET     (AD) TO LEFT BRACKET     (4A)
* MAP GREATER OR EQUAL(AE) TO RIGHT CHEVRON     (8B)
         DC    X'6D90A2A3A4A5A6A7A8A9AA4B4B4A8BAF'  /*A0 - AF*//*@D1C*/
* MAP SUBSCRIPTS 0-9 (B0-9) TO NUMERICS 0-9  (F0-9)
* MAP BOX CNR-LWR RGHT(BB) TO PERIOD            (4B)
* MAP BOX CNR-UPR RGHT(BC) TO PERIOD            (4B)
* MAP RIGHT BRACKET    (BD) TO RIGHT BRACKET    (5A)
* MAP NOT EQUAL        (BE) TO HASH             (7B)
* MAP LONG UNDERSCORE (BF) TO UNDERSCORE        (6D)
         DC    X'F0F1F2F3F4F5F6F7F8F9BA4B4B5A6F6D'  /*B0 - BF*//*@D1C*/
         DC    X'C0C1C2C3C4C5C6C7C8C9CACBCCCDCECF'  /*C0 - CF*//*@D1C*/
         DC    X'D0D1D2D3D4D5D6D7D8D9DADBDCDDDEDF'  /*D0 - DF*//*@D1C*/
         DC    X'E0E1E2E3E4E5E6E7E8E9EAEBECEDEEEF'  /*E0 - EF*//*@D1C*/
         DC    X'F0F1F2F3F4F5F6F7F8F9FAFBFCFDFEFF'  /*F0 - FF*//*@D1C*/
         EJECT
DSVS6PFC DS    0XL256       THE PRINT FIDELITY TABLE
*
*                             X'00' - PRINT FIDELITY MAINTAINED
*                             X'04' - PRINT FIDELITY COMPROMISED
*
*  NOTE  NOTE  NOTE  NOTE  NOTE  NOTE  NOTE  NOTE  NOTE  NOTE  NOTE
*
*  THERE IS A DISOSS PROBLEM OUTSTANDING AT THE DATE OF THE LATEST
*  REVISION OF THIS TABLE (JAN 24 1984). IF THE TABLE INDICATES
*  THAT PRINT FIDELITY HAS BEEN COMPROMISED, DISOSS WILL DELIVER A
*  GARBAGE DOCUMENT TO DISOSS/PS. DISOSS/PS WILL SHOW AN ENTRY IN THE
*  MAIL LOG, BUT WHEN THE DOCUMENT IS DISPLAYED NOTHING WILL APPEAR IN
*  THE AREA OF THE SCREEN WHERE THE TEXT SHOULD BE.
*  IF THE PROBLEM HAS BEEN FIXED BY THE TIME YOU INSTALL THIS TABLE,
*  YOU SHOULD DETERMINE WHICH OF THE TRANSLATIONS IN THE ABOVE TABLE
*  CAUSE A LOSS OF PRINT FIDELITY, AND MODIFY THE FOLLOWING TABLE
*  ACCORDINGLY. TO DO THIS, ENTER X'04' IN THE POSITION IN THE TABLE
*  WHICH CORRESPONDS TO THE HEX VALUE OF THE INPUT CHARACTER THAT
*  TRANSLATES TO A DIFFERENT SYMBOL ON OUTPUT. FOR EXAMPLE, THE LOWER
*  LEFT BOX CORNER CHARACTER (X'AB') HAS NO EQUIVALENT IN THE OUTPUT
*  GCID AND WE HAVE CHOSEN TO TRANSLATE IT TO X'4B' WHICH IS A PERIOD
*  (FULL STOP) ON THE MULTILINGUAL CODE PAGE. THEREFORE A VALUE OF
*  X'04' SHOULD BE PLACED IN THE TWELFTH ENTRY OF THE ELEVENTH LINE
*  OF THE TABLE THUS:
*
*        DC    X'00000000000000000000000400000000'  /*A0 - AF*//*@D1C*/
*
*  NOTE HOWEVER THAT NOT ALL OF THE ENTRIES IN THE TRANSLATE TABLE
*  WHICH ALTER THE HEX VALUE OF THE OUTPUT CHARACTER CAUSE A LOSS OF
*  FIDELITY. FOR EXAMPLE, A CENT SYMBOL IS X'4A' ON THE INPUT. THIS
*  TRANSLATES TO X'B0' ON OUTPUT. X'B0' IN THE MULTILINGUAL CODE
*  PAGE IS A CENT SYMBOL, SO THE INPUT AND OUTPUT SYMBOLS ARE THE
*  SAME. THE INPUT VALUES WHICH TRANSLATE TO A DIFFERENT SYMBOL USING
*  THE ABOVE TABLE ARE 8C,9C,9F,A0,AB,AC,AE,B0-B9,BB,BC,BE,AND BF.
*
         DC    X'00000000000000000000000000000000'  /* 00 - 0F */
         DC    X'00000000000000000000000000000000'  /* 10 - 1F */
         DC    X'00000000000000000000000000000000'  /* 20 - 2F */
         DC    X'00000000000000000000000000000000'  /* 30 - 3F */
         DC    X'00000000000000000000000000000000'  /* 40 - 4F */
         DC    X'00000000000000000000000000000000'  /*50 - 5F*//*@D1C*/
         DC    X'00000000000000000000000000000000'  /*60 - 6F*//*@D1C*/
         DC    X'00000000000000000000000000000000'  /*70 - 7F*//*@D1C*/
         DC    X'00000000000000000000000000000000'  /*80 - 8F*//*@D1C*/
         DC    X'00000000000000000000000000000000'  /*90 - 9F*//*@D1C*/
         DC    X'00000000000000000000000000000000'  /*A0 - AF*//*@D1C*/
         DC    X'00000000000000000000000000000000'  /*B0 - BF*//*@D1C*/
         DC    X'00000000000000000000000000000000'  /*C0 - CF*//*@D1C*/
         DC    X'00000000000000000000000000000000'  /*D0 - DF*//*@D1C*/
         DC    X'00000000000000000000000000000000'  /*E0 - EF*//*@D1C*/
         DC    X'00000000000000000000000000000000'  /*F0 - FF*//*@D1C*/
         END
/*
//
```

A.4 IMPROVEMENTS AND ALTERNATIVES

This section is a list of ideas for programming improvements and alternative options.

- DBTBAT1 performs the entire construction of its output 80-byte header. On reflection, it would probably be better for it to copy the input header into the output header, modifying only those fields concerning DBTBAT1 directly. The philosophy of this is to increase efficiency, to minimise the processing of fields which pass through a program without being used or modified, and to decrease the likely number of changes should the meaning of an area be altered.

- More analysis could be made of the responses from the OPEN/CLOSE in the program DBTMOV1, and the response from DISOSS to the programs DBTSON1, DBTSOF1, DBTCLN1, and DBTSND1. Errors could be identified instead of just detected.

  Most of the above CICS programs issue an API queue PURGE command upon error detection. This deletes all API requests associated with the issuing user. This is safe for the DISOSS system, but somewhat drastic for the user, especially since currently the mailbox API requests are all issued in the name of the same user. Some errors may not require the purge.

- The initial data inserted into the DCA datastream by DBTTRN1 from structure DCAINI could be modified according to input profile information such as the pagelength and pagewidth. DBTTRN1 could also be made to deduce information about the document. Some deduced information might only be available to the program after the body of the document has been processed. For example the pagelength or pagewidth might be specified by the user as undefined, causing DBTTRN1 to keep a tally on the largest output lines as they are created. Such information would then have to be written to the profile area in the first and last output records using a READ UPDATE.

- Some users may wish to turn the facility on and off. The facility can be turned on, but it cannot be safely turned off, because irreversible damage appears to be done to the API queue by a Request_Distribution without a preceding Sign_On. Perhaps the simplest way to switch the system off would be to use CEMT to disable transaction DBTM.

A.5 SIMULTANEOUS CICS/BATCH ACCESS TO SHARED DATASET

A CICS attempt to open dataset DBTVSQ0 during batch access caused DBTM to stop, and reinitiate itself, as intended, and allowed the batch job to continue normal execution.

A batch attempt to open dataset DBTVSQ0 during CICS access did not terminate the job, allowed DBTM to continue normal execution, and caused IEC161 052-084 data management messages to be displayed on the MVS operator console until CICS relinquished control. In a production environment, it would be worth warning the operators that these messages are expected and normal.

B.0 API AND DIU-BUILD SUBROUTINES

The DIU-building and API interface utilities were written by Martin Hibbert of IBM UK Technical Support, and are documented in **DISOSS Application Interface: Programming Guidelines, GG24-1614**. A few of them have been modified for use in our system; these are identified by names ending in '2', e.g. APIDPR2, APIDPRP2, APIRETP2, APIDIS2, APIFIL2, APIDIUS2.

B.1 ASSEMBLER CONTROL BLOCKS

B.1.1 APICOM

DISOSS API program common area

```
************************************************************************
*  APICOM : DISOSS API PROGRAM COMMON AREA                             *
************************************************************************
         SPACE
APICOM   DSECT
         SPACE
COMXN    DC    CL8'*APICOM*'         EYE CATCHER NAME
COMXC    DC    CL8'COMMAND'           & CURRENT COMMAND
COMXT    DC    CL4'TRAN' '            & TRANSACTION ID
         SPACE
COMPARMS DS    0F                 STANDARD PARM LIST:
COMP1    DS    A
COMP2    DS    A
COMP3    DS    A
COMP4    DS    A
COMP5    DS    A
COMP6    DS    A
COMP7    DS    A
         SPACE
COMCPTR  DS    A                     POINTER TO CONTROL BLOCK
COMRETCD DS    F                     RETURN CODE
COMREASN DS    F                     REASON CODE
COMDPTR  DS    A                     POINTER TO COMMAND DATA
COMDLEN  DS    H                     DATA LENGTH
COMSPAR1 DS    H                     SPARE
         SPACE
COMDPRP  DS    A                     DOC PROFILE POINTER
COMDIUP  DS    A                     OUTPUT DIU POINTER
         SPACE
COMDBUFP DS    A                     INTERNAL DIU BUFFER POINTER
COMLDIU1 DS    H                     TOTAL DIU DATA LEGTH
COMLDIU2 DS    H                     TOTAL DIU SEGMENT LENGTH
COMLDIU3 DS    H                     AMOUNT OF SEGMENT USED SO FAR
COMLDIU4 DS    H
COMDCMD  DS    X                     WHAT COMMAND IS BEING SENT
COMDDLV  EQU   X'01'                 THIS IS DELIVER COMMAND
COMDACK  EQU   X'02'                 THIS IS ACK
COMDSON  EQU   X'03'                 THIS IS SIGNON RESPONSE
```

```
COMDST   DS    B                   STATUS FLAGS WITHIN DIU ANALYSIS
COMDSTC  EQU   X'01'                COMMAND + OPERANDS ANALYSED
COMDSTP  EQU   X'02'                DOCUMENT PROFILE ANALYSED
COMDSTD  EQU   X'04'                DOCUMENT TEXT BUFFER FULL
COMDSTX  EQU   X'40'                HAVE REACHED SUFFIX
COMDSTE  EQU   X'80'                HAVE REACHED END OF DATA
         SPACE
COMTRFLG DS    C                    DO EXTRA TRACE CALLS IF "T"
COMTRYES EQU   C'T'
COMTRREG DS    16F                  SAVE AREA WHEN DOING TRACE
         SPACE
COMLEN   EQU   *-APICOM          LENGTH OF API COMMON AREA
```

B.1.2 APIDPR2

This is a modification of the original APIDPR.

```
***************************************************************
* AREA FOR COMMAND PARAMETERS AND DOCUMENT PROFILE            *
***************************************************************
         SPACE
APIDPR   DSECT
DPR      DS    0F
         SPACE
* FOLLOWING ARE RESERVED FOR COMPATIBILITY WITH TEXT 4K BLOCKS
DPRRESV  EQU   *
         DS    H
         DS    H
         DS    F
DPRRESVL EQU   *-DPRRESV
         SPACE
* COMMAND PARAMETERS SECTION.
DPRCMD   DS    0F
         SPACE
* 'DELIVER' COMMAND
DPRSID   DS    CL8                 SOURCE ID
DPRRID   DS    CL8                 RECIPIENT ID
DPRDDN   DS    CL8                 DDN (LOCATION)
DPRDIS   DS    CL20                DISTN DOC NAME
DPRMSG   DS    CL255               MESSAGE
DPRMSGL  DS    AL1                  & LENGTH
         SPACE
* 'ACKNOWLEDGE' COMMAND
         ORG   DPRCMD
DPREXCOD DS    CL3                 ACK EXCEPTION CODE
DPRACKR  EQU   *                   ACK REPLY
DPRACKRF DS    CL8                 FILE: DTM PART OF LADN
         ORG   DPRACKR
DPRACKRD DS    CL20                DISTRIB: DISTN DOC NAME
         ORG   DPRACKR
DPRACKRS DS    CL7                 SEARCH: COUNT VALUE
         ORG
         SPACE
* END OF COMMAND PARAMETERS SECTION
DPRCMDL  EQU   *-DPRCMD            LENGTH OF COMMAND PARAMETERS
         SPACE
* DOC PROFILE PART
DPRPROF  EQU   *
DPRDOT   DS    H                   DOC TYPE
DPRSYS   DS    CL13                SYSTEM ID VALUE
DPRPGC   DS    F                   PROFILE GCID
DPRLAD   DS    CL8                 LADN  (DTM PART ONLY)
DPRACC   DS    CL4                 ACCESS CODE VALUE
DPRACCL  DS    AL1                  AND LENGTH
DPRDON   DS    CL15                DOCUMENT NAME
DPRDONL  DS    AL1                  & LENGTH
DPRSUB   DS    CL60                SUBJECT
DPRSUBL  DS    AL1                  & LENGTH
DPRAUT   DS    CL60                AUTHORS
DPRAUTL  DS    AL1                  & LENGTH
DPRREC   DS    CL60                RECIPIENTS
```

```
DPRRECL  DS    AL1                  & LENGTH
DPRKEY   DS    CL60                KEYWORDS (SEARCH TERMS)
DPRKEYL  DS    AL1                  & LENGTH
DPRCOR   DS    CL16                CORRELATION DATA
DPRCORL  DS    AL1                  & LENGTH
DPRPROFL EQU   *-DPRPROF           LENGTH OF PROFILE SECTION
         SPACE
DPRLEN   EQU   *-DPR            LENGTH OF DPR DSECT
```

B.1.3 APIRET

```
************************************************************************
*  DEFINE THE STRUCTURE RETURNED BY RETRIEVE                           *
************************************************************************
         SPACE
APIRET   DSECT                     AREA FOR CICS RETRIEVE
RET      DS    0F
RETAPID  DC    CL4'APID'           IDENTIFIER (WE HOPE)
RETDIUID DS    CL16                DIU ID OF ORIGINAL REQUEST DIU
RETKEY   DS    0CL16               KEY VALUE TO PASS ON ACTIVATE
RETNAME  DS    CL8                 USER NAME
RETTIME  DS    CL8                 TIME STAMP
RETLEN   EQU   *-RET
         SPACE
```

B.1.4 APIREGS

```
         SPACE
************************************************************************
*                                                                      *
*  RISC SAMPLE DISOSS API ROUTINES - STANDARD REGISTER EQUATES         *
*                                                                      *
************************************************************************
         SPACE
R0       EQU   0                   USED TO HOLD LENGTHS ETC.
R1       EQU   1                   USED TO HOLD ADDRESSES
R2       EQU   2
R3       EQU   3                   MODULE BASE
R4       EQU   4
R5       EQU   5
R6       EQU   6
R7       EQU   7
R8       EQU   8                   ADDRESSES COMMON AREA
R9       EQU   9                   API CONTROL BLOCK POINTER
R10      EQU   10
R11      EQU   11                  EIB ADDRESS
R12      EQU   12
R13      EQU   13                  DYNAMIC STORAGE REGISTER
R14      EQU   14
R15      EQU   15
         SPACE
```

B.2 PL/I CONTROL BLOCKS

B.2.1 APICOMP

```
/********************************************************************/
/* (APICOMP) - API COMMON AREA - PLI VERSION - API SAMPLE PROGRAMS */
/********************************************************************/

DCL COMPTR POINTER;

DCL
 1 APICOM BASED(COMPTR),

   2 COMXN CHAR(8),        /* EYE CATCHER NAME                      */
   2 COMXC CHAR(8),        /* DISOSS FUNCTION CODE                  */
   2 COMXT CHAR(4),        /* TRANSACTION ID                        */

   2 COMPARMS,             /* USED IN ASSEMBLER SUBROUTINES         */
     3 COMP1 POINTER,      /* CAN BE USED IN MAINLINE               */
     3 COMP2 POINTER,
     3 COMP3 POINTER,
     3 COMP4 POINTER,
     3 COMP5 POINTER,
     3 COMP6 POINTER,
     3 COMP7 POINTER,

   2 COMCPTR POINTER,      /* DISOSS API CONTROL BLOCK POINTER      */
   2 COMRETCD FIXED BIN(31), /* RETURN CODE VALUE                   */
   2 COMREASN FIXED BIN(31), /* REASON CODE VALUE                   */
   2 COMDPTR POINTER,        /* DATA POINTER                        */
   2 COMDLEN FIXED BIN(15),  /* DATA LENGTH                         */
   2 COMSPAR1 FIXED BIN(15), /* SPARE HALFWORD                      */

   2 COMDPRP POINTER,      /* DPR BLOCK POINTER                     */
   2 COMDIUP POINTER,      /* DIU BUFFER POINTER                    */

   2 COMPARSE,             /* AREA USED BY PARSER ROUTINE           */
    3 COMDBUFP POINTER,
    3 COMLDIU1 FIXED BIN(15),
    3 COMLDIU2 FIXED BIN(15),
    3 COMLDIU3 FIXED BIN(15),
    3 COMLDIU4 FIXED BIN(15),
    3 COMDCMD BIT(8),                 /* COMMAND CODE               */
    3 COMDST CHAR(1),      /* PARSING STATUS BYTE                   */

   2 COMTRFLG CHAR(1),     /* TRACE FLAG. IF "T" THEN ALL DISOSS    */
                           /* CALLS WILL BE TRACED                  */
   2 COMTRSP CHAR(1),      /* FOR ALIGNMENT */
   2 COMTRREG(16) FIXED BIN(31); /* USED BY ASSEMBLER ROUTINES      */

DCL
  COMDDLV BIT(8) INIT('00000001'B),
  COMDACK BIT(8) INIT('00000010'B),
  COMDSON BIT(8) INIT('00000011'B);

DCL
  COMTRYES CHAR(1) INIT('T');
```

B.2.2 APIDPRP

Profile and command parameter area - PL/I. Used by programs DBTSON1 and DBTSOF1.

```
DCL
 1 APIDPR BASED(COMDPRP),
   3 DPRRESV,
     5 DPRRESV1 FIXED BIN(15),
     5 DPRRESV2 FIXED BIN(15),
     5 DPRRESV3 FIXED BIN(31),

   3 DPRCMD CHAR(300),

   3 DPRPROF,
     5 DPRDOT FIXED BIN(15),
     5 DPRSYS CHAR(13),
     5 DPRPGC FIXED BIN(31),
     5 DPRLAD CHAR(8),
     5 DPRACC CHAR(4),
     5 DPRACCL BIT(8),
     5 DPRDON CHAR(15),
     5 DPRDONL BIT(8),
     5 DPRSUB CHAR(60),
     5 DPRSUBL BIT(8),
     5 DPRAUT CHAR(60),
     5 DPRAUTL BIT(8),
     5 DPRREC CHAR(60),
     5 DPRRECL BIT(8),
     5 DPRKEY CHAR(60),
     5 DPRKEYL BIT(8);
DCL
 1 DPRCMD1 BASED(COMDPRP),
     3 DPRRESV1 FIXED BIN(15),
     3 DPRRESV2 FIXED BIN(15),
     3 DPRRESV3 FIXED BIN(31),
   3 DPRSID CHAR(8),
   3 DPRRID CHAR(8),
   3 DPRDDN CHAR(8),
   3 DPRDIS CHAR(20),
   3 DPRMSG CHAR(255),
   3 DPRMSGL BIT(8);
DCL
 1 DPRCMD2 BASED(COMDPRP),
     3 DPRRESV1 FIXED BIN(15),
     3 DPRRESV2 FIXED BIN(15),
     3 DPRRESV3 FIXED BIN(31),
   3 DPREXCOD BIT(18),
   3 DPRACKR  CHAR(20),
 DPRACKRF CHAR(8)  BASED(ADDR(DPRACKR)),
 DPRACKRD CHAR(20) BASED(ADDR(DPRACKR)),
 DPRACKRS CHAR(7)  BASED(ADDR(DPRACKR));
```

B.2.3 APIDPRP2

Profile and command parameter area - PL/I. Used by CICS program DBTSND1. This is a modification of the original APIDPRP.

```
 DCL
  1 APIDPR BASED(COMDPRP),
    3 DPRRESV,
      5 DPRRESV1 FIXED BIN(15),
      5 DPRRESV2 FIXED BIN(15),
      5 DPRRESV3 FIXED BIN(31),

    3 DPRCMD CHAR(300),

    3 DPRPROF,
      5 DPRDOT FIXED BIN(15),
      5 DPRSYS CHAR(13),
      5 DPRPGC FIXED BIN(31),
      5 DPRLAD CHAR(8),
      5 DPRACC CHAR(4),
      5 DPRACCL BIT(8),
      5 DPRDON CHAR(15),
      5 DPRDONL BIT(8),
      5 DPRSUB CHAR(60),
      5 DPRSUBL BIT(8),
      5 DPRAUT CHAR(60),
      5 DPRAUTL BIT(8),
      5 DPRREC CHAR(60),
      5 DPRRECL BIT(8),
      5 DPRKEY CHAR(60),
      5 DPRKEYL BIT(8),
      5 DPRCOR CHAR(16),
      5 DPRCORL BIT(8);
 DCL
  1 DPRCMD1 BASED(COMDPRP),
      3 DPRRESV1 FIXED BIN(15),
      3 DPRRESV2 FIXED BIN(15),
      3 DPRRESV3 FIXED BIN(31),
    3 DPRSID CHAR(8),
    3 DPRRID CHAR(8),
    3 DPRDDN CHAR(8),
    3 DPRDIS CHAR(20),
    3 DPRMSG CHAR(255),
    3 DPRMSGL BIT(8);
 DCL
  1 DPRCMD2 BASED(COMDPRP),
      3 DPRRESV1 FIXED BIN(15),
      3 DPRRESV2 FIXED BIN(15),
      3 DPRRESV3 FIXED BIN(31),
    3 DPREXCOD BIT(18),
    3 DPRACKR  CHAR(20),
  DPRACKRF CHAR(8)  BASED(ADDR(DPRACKR)),
  DPRACKRD CHAR(20) BASED(ADDR(DPRACKR)),
  DPRACKRS CHAR(7)  BASED(ADDR(DPRACKR));
```

B.2.4 APIRETP

API response start data area. Used by programs DBTSON1 and DBTSOF1.

```
DCL RETPTR POINTER;
DCL 1 APIRET BASED(RETPTR),
      2 RETAPID CHAR(4),
      2 RETDIUID CHAR(2),
      2 RETDIUI2 CHAR(14),
      2 RETNAME CHAR(8),
      2 RETTIME CHAR(8);
DCL RETSON CHAR(2) INIT('01'),
    RETOBT CHAR(2) INIT('02'),
    RETFIL CHAR(2) INIT('03'),
    RETSOF CHAR(2) INIT('04');
```

B.2.5 APIRETP2

API response start data area. Used by program DBTRSP1. This is a modification of the original APIRETP.

```
DCL RETPTR POINTER;
DCL 1 APIRET BASED(RETPTR),
      2 RETAPID CHAR(4),
      2 RETDIUID CHAR(16),
      2 RETNAME CHAR(8),
      2 RETTIME CHAR(8);
DCL RETSON CHAR(2) INIT('01'),
    RETOBT CHAR(2) INIT('02'),
    RETFIL CHAR(2) INIT('03'),
    RETSOF CHAR(2) INIT('04');
```

B.3 ASSEMBLER MACRO

B.3.1 APICALL

This is the macro used for subroutine calls.

```
         MACRO
&NAME    APICALL  &ENTRY,&OPRNDS
         LCLA  &N,&C,&A
&N       SETA  N'&OPRNDS
&C       SETA  0
         AIF   ('&NAME' EQ '').NONAME
&NAME    DS    0F
.NONAME  ANOP
         L     R0,DFHEIBP          LOAD EIB POINTER
         ST    R0,COMP1            STORE IN PARM LIST
         LA    R0,APICOM           LOAD ADDR OF APICOM
         ST    R0,COMP2            STORE IN PARM LIST
.LOOP    ANOP
&C       SETA  &C+1
         AIF   (&C GT &N).BAL
&A       SETA  &C+2
         LA    R0,&OPRNDS(&C)      ADDRESS OF PARM
         ST    R0,COMP&A           STORE IN LIST
         AGO   .LOOP
.BAL     ANOP
         LA    R1,COMPARMS         ADDRESS OF PARAMETERS
         L     15,=V(&ENTRY.)      LOAD 15 WITH ENTRY ADR
         BALR  14,15               BRANCH TO ENTRY POINT
         L     R15,COMRETCD        RETURN CODE IN R15
         L     R0,COMREASN         & REASON CODE IN R0
         MEND
```

B.4 SUBROUTINE LISTINGS

B.4.1 APIACTIV

```
ACT  TITLE '*APIACTIV* - PERFORM DISOSS "ACTIVATE" - DISOSS API SAMPLE *
               PROGRAMS'
          PRINT NOGEN
APIACTIV  CSECT
***********************************************************************
*                                                                     *
*   SUBROUTINE TO DO A DISOSS 'ACTIVATE'                              *
*                                                                     *
*   INPUT PARMLIST:                                                   *
*                    -> NAME (8 BYTES)                                *
*                    -> TIME STAMP (8 BYTES)                          *
*                                                                     *
*   COMCPTR IS SET TO ZERO BY CALLER IF THIS IS A FIRST TIME CALL     *
*                                                                     *
*   OUTPUT: COMRETCD HAS RETURN CODE.                                 *
*             = 8 : BAD RETURN FROM DISOSS CALL.                      *
*                   DISOSS RC IN COMREASN                             *
*                                                                     *
***********************************************************************
         SPACE
         L     R8,4(R1)            APICOM ADDRESS
         USING APICOM,R8
         SPACE
         XC    ACTRSV,ACTRSV       ZERO RESERVED FIELD
         L     R4,8(R1)            GET NAME POINTER
         MVC   ACTNAME,0(R4)       MOVE IN USER NAME
         L     R4,12(R1)           ADDR OF TIME DATA
         MVC   ACTTIME,0(R4)       MOVE INTO STD PARMS
         SPACE
         LA    R0,ACT              GET COMMAND DATA
         ST    R0,COMDPTR           & TELL DISOSS
         LA    R0,ACTLEN           GET DATA LENGTH
         STH   R0,COMDLEN           TELL DISOSS
         SPACE
         APICALL APIDISOS,(=CL8'ACTIVATE')
         SPACE
* WE RETURN WITH THE RETURN CODE FROM DISOSS
         ST    R15,COMREASN       OUR REASON CODE IS DISOSS RC
         LTR   R15,R15            WAS IT BAD RC ?
         BZ    *+8                NO, USE ZERO AS OUR RC
         LA    R15,8              OTHERWISE WE HAVE RC 8
         ST    R15,COMRETCD
         EJECT
DFHEISTG DSECT
ACT      DS    0F                  ACTIVATE PARAMETER DATA
ACTNAME  DS    CL8'USERNAME'       USER NAME GOES HERE
ACTTIME  DS    XL8                 TIME VALUE
ACTRSV   DS    XL4                 RESERVED
ACTLEN   EQU   *-ACT
         SPACE
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR
COPY  APIREGS
PRINT ON
END
```

B.4.2 APIDIUSB

This is a collection of DIU element building subroutines. Used by program DBTSON1 via routine APISGNON.

```
************************************************************************
*  SUBROUTINES TO BUILD DIU ELEMENTS                                   *
************************************************************************
*  REGISTERS ARE NOT SAVED IN THESE ROUTINES.                         *
*  REGS 0,1 CARRY PARAMETER VALUES AND ARE ALSO USED AS WORK REGS.    *
*  REG 15 IS OCCASIONALLY USED AS A WORK REGISTER                      *
*  REG 2 POINTS TO THE CURRENT POSITION IN THE OUTPUT BUFFER           *
************************************************************************
         SPACE
************************************************************************
*  PREFIX AND SUFFIX BUILDERS                                          *
************************************************************************
         SPACE
DIUPFX   DS    0F
         LR    R15,R0                     LENGTH OF CORRELATION DATA
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C00102'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCPFX                 MOVE IN CORRELATION DATA
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCPFX   MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE 2
DIUSFX   DS    0F
         MVC   2(3,R2),=X'CF0100'         STANDARD SUFFIX DATA
         LA    R0,5                       FIXED LENGTH
         STH   R0,0(R2)                    OF FIVE
         LA    R2,5(R2)                   INCREMENT BUFFER POINTER
         BR    R14                         & RETURN
         SPACE 2
************************************************************************
*  COMMANDS                                                            *
************************************************************************
         SPACE
FILECMD  DS    0F              BUILD A DIU 'FILE' COMMAND
*                                         LL ADDED LATER
         MVC   2(3,R2),=X'CC0201'         MOVE CTF
         LA    R2,5(R2)                   INCR BUFFER POINTER
         BR    R14                        RETURN TO CALLER
         SPACE 2
SONCMD   DS    0F              BUILD A DIU 'SIGNON' COMMAND
*                                         LL ADDED LATER
         MVC   2(3,R2),=X'CD0C01'         MOVE CTF
         LA    R2,5(R2)                   INCR BUFFER POINTER
         BR    R14                        RETURN TO CALLER
         SPACE 2
OBTCMD   DS    0F              BUILD A DIU 'OBTAIN' COMMAND
*                                         LL ADDED LATER
         MVC   2(3,R2),=X'CC1701'         MOVE CTF
         LA    R2,5(R2)                   INCR BUFFER POINTER
         BR    R14                        RETURN TO CALLER
         SPACE 2
DISCMD   DS    0F              BUILD A DIU 'DISTRIBUTE' COMMAND
*                                         LL ADDED LATER
         MVC   2(3,R2),=X'CC1C01'         MOVE CTF
         LA    R2,5(R2)                   INCR BUFFER POINTER
         BR    R14                        RETURN TO CALLER
         SPACE 2
************************************************************************
*  PARAMETERS FOR 'FILE' & 'DISTRIBUTE'                                *
************************************************************************
         SPACE
IDDATA1  DS    0F
         MVC   2(3,R2),=X'C52001'         MOVE CTF
         STC   R0,5(R2)                   IDENTIFIED DATA IN R0
         LA    R0,6
         STH   R0,0(R2)                   SET LL
         LA    R2,6(R2)                   INCR BUFFER POINTER
         BR    R14
         SPACE 2
ACCCODE  DS    0F
         MVC   2(3,R2),=X'C33941'         SET CTF VALUE
         MVC   5(2,R2),=X'0601'           LENGTH + TYPE
         L     R0,0(R1)                   LOAD ACCESS CODE VALUE
         ST    R0,7(R2)                   SET ACCESS CODE VALUE
         LA    R0,11
         STH   R0,0(R2)                   SET LL VALUE
         LA    R2,11(R2)                  UPDATE TOTAL LENGTH
         BR    R14
         SPACE 2
DUSEGN   DS    0F
*                                         LL SET LATER
         MVC   2(6,R2),=X'C90381200000'   SEGMENT INTRODUCER
         LA    R2,8(R2)                   INCREMENT BUFFER PTR
         BR    R14
         SPACE 2
DUSEGL   DS    0F
*                                         LL SET LATER
         MVC   2(6,R2),=X'C90381000000'   SEGMENT INTRODUCER
         LA    R2,8(R2)                   INCREMENT BUFFER PTR
         BR    R14
         SPACE 2
DUSYS    DS    0F
*                                         LL SET LATER
         STH   R0,0(R2)                   SET DOC TYPE
         MVC   2(13,R2),0(R1)             MOVE SYSTEM ID
         LA    R2,15(R2)                  INCR BUF PTR
         BR    R14
         SPACE 2
PRDOC    DS    0F
*                                         LL SET LATER
         MVC   2(3,R2),=X'CA0301'         CTF
         LA    R2,5(R2)                   INCR BUF PTR
         BR    R14
         SPACE 2
PRBASE   DS    0F
*                                         LL SET LATER
         MVC   2(3,R2),=X'CA0401'         CTF
         LA    R2,5(R2)                   INCR BUF PTR
         BR    R14
         SPACE 2
PRBTYPE  DS    0F
         MVC   2(3,R2),=X'C70601'          CTF
         STH   R0,5(R2)                    DOC TYPE
         LA    R0,7
         STH   R0,0(R2)                    SET LL VALUE
         LA    R2,7(R2)                    INCR BUF PTR
         BR    R14
         SPACE 2
PRBGCID  DS    0F
         MVC   2(3,R2),=X'C70101'          CTF
         ST    R0,5(R2)                    SET GCID/GPID
         LA    R0,9
         STH   R0,0(R2)                    SET LL VALUE
         LA    R2,9(R2)                    INCR BUF PTR
         BR    R14
         SPACE 2
PRBDOCN  DS    0F
         LR    R15,R0                      LENGTH OF DOC NAME
         LA    R0,5(R15)                   ADD 5 FOR LLCTF
         STH   R0,0(R2)                    STORE LL VALUE
         MVC   2(3,R2),=X'C70001'          MOVE CTF DATA
         BCTR  R15,0                       -1 FOR EXECUTE
         EX    R15,MVCPRBD                 MOVE IN DOC NAME
         LA    R2,6(R15,R2)                POINT AT NEXT AVAIL BYTE
         BR    R14                         & RETURN
         SPACE
MVCPRBD  MVC   5(1,R2),0(R1)               EXECUTED MOVE
         SPACE
PRBSUBJ  DS    0F
         LR    R15,R0                      LENGTH OF SUBJECT FIELD
         LA    R0,5(R15)                   ADD 5 FOR LLCTF
         STH   R0,0(R2)                    STORE LL VALUE
         MVC   2(3,R2),=X'C70B01'          MOVE CTF DATA
         BCTR  R15,0                       -1 FOR EXECUTE
         EX    R15,MVCPRBS                 MOVE IN SUBJECT
         LA    R2,6(R15,R2)                POINT AT NEXT AVAIL BYTE
         BR    R14                         & RETURN
         SPACE
MVCPRBS  MVC   5(1,R2),0(R1)               EXECUTED MOVE
         SPACE
PRBAUTH  DS    0F
         LR    R15,R0                      LENGTH OF AUTHOR FIELD
         LA    R0,5(R15)                   ADD 5 FOR LLCTF
         STH   R0,0(R2)                    STORE LL VALUE
         MVC   2(3,R2),=X'C70401'          MOVE CTF DATA
         BCTR  R15,0                       -1 FOR EXECUTE
         EX    R15,MVCPRBA                 MOVE IN AUTHOR NAME
         LA    R2,6(R15,R2)                POINT AT NEXT AVAIL BYTE
         BR    R14                         & RETURN
         SPACE
MVCPRBA  MVC   5(1,R2),0(R1)               EXECUTED MOVE
         SPACE
PRBTIME  DS    0F
         MVC   2(3,R2),=X'C70701'          CTF VALUE
         MVC   5(6,R2),DATETIME            STD VALUE FOR THE MOMENT
         LA    R0,11                       LENGTH
         STH   R0,0(R2)
         LA    R2,11(R2)                   POINT TO NEXT FREE BYTE
         BR    R14
         SPACE
DATETIME DC    AL2(1983),AL1(05),AL1(21),AL1(22),AL1(30)
* 22.30 ON 21ST MAY 1983
         SPACE
***********************************************************************
*  PARAMETERS FOR 'SIGNON'                                            *
***********************************************************************
         SPACE
FUNCSETS DS    0F
         LR    R15,R0                     LENGTH OF NAME FIELD
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C31201'         CTF VALUE
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCFUNC                MOVE IN FUNC SETS STRING
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCFUNC  MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
SONNAME  DS    0F
         LR    R15,R0                     LENGTH OF NAME FIELD
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C30D01'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCSONN                MOVE IN USER NAME
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCSONN  MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
SONPASS  DS    0F
         LR    R15,R0                     LENGTH OF PASSWORD
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C33801'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCSONP                MOVE IN PASSWORD
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCSONP  MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
RCVDOCS  DS    0F                        SINGLE RECEIVE DOC TYPE
         MVC   2(3,R2),=X'C32901'         CTF
         STH   R0,5(R2)                   STORE DOC TYPE
         LA    R0,7                       LENGTH
         STH   R0,0(R2)                   STORED IN LL FIELD
         LA    R2,7(R2)                   BUMP BUFPTR
         BR    R14
         SPACE
***********************************************************************
*  PARAMETERS FOR 'OBTAIN'                                            *
***********************************************************************
         SPACE
OBTOPTS  DS    0F                         OBTAIN-OPTIONS
         MVC   2(3,R2),=X'C31E01'         CTF
         STC   R0,5(R2)                   STORE OPTION BYTE
         LA    R0,6                       LENGTH
         STH   R0,0(R2)                   STORED IN LL FIELD
         LA    R2,6(R2)                   BUMP BUFPTR
         BR    R14
         SPACE
************************************************************************
*  PARAMETERS FOR 'DISTRIBUTE'                                         *
************************************************************************
         SPACE
DNADDR   DS    0F                   DESTINATION NODE ADDRESS
         LR    R15,R0                     LENGTH OF ADDRESS NAME
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C32F01'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCDNAD                MOVE IN ADDRESS
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCDNAD  MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
RECADDR  DS    0F                   RECIPIENT NAME
         LR    R15,R0                     LENGTH OF NAME
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C30601'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCRECAD               MOVE IN NAME
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCRECAD MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
************************************************************************
*  DOCUMENT BUILDERS                                                   *
************************************************************************
         SPACE
DCI      DS    0F
         MVC   2(3,R2),=X'CB0101'
         LA    R0,5
         STH   R0,0(R2)
         LA    R2,5(R2)
         BR    R14
```

B.4.3 APIDIUS2

This collection of DIU element building subroutines is a modified version of APIDIUSB, and is used by program DBTSND1 via routines APIDIS2 and APIFIL2.

```
************************************************************************
*  SUBROUTINES TO BUILD DIU ELEMENTS                                   *
************************************************************************
*  COPY OF APIDIUSB, ADDED OPERANDS FOR                                *
*                LENGTH OF CORRELATION DATA 16 BYTES (INSTEAD OF 2),   *
*                SOURCE ADDRESS,                                       *
*                DOCUMENT GCID (NOT USED IN DBT... SYSTEM),            *
*                ATTRIBUTE LIST                                        *
************************************************************************
*  REGISTERS ARE NOT SAVED IN THESE ROUTINES.                          *
*  REGS 0,1 CARRY PARAMETER VALUES AND ARE ALSO USED AS WORK REGS.     *
*  REG 15 IS OCCASIONALLY USED AS A WORK REGISTER                      *
*  REG 2 POINTS TO THE CURRENT POSITION IN THE OUTPUT BUFFER           *
************************************************************************
         SPACE
************************************************************************
*  PREFIX AND SUFFIX BUILDERS                                          *
************************************************************************
         SPACE
DIUPFX   DS    0F
         LR    R15,R0                     LENGTH OF CORRELATION DATA
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C00102'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCPFX                 MOVE IN CORRELATION DATA
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCPFX   MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE 2
DIUSFX   DS    0F
         MVC   2(3,R2),=X'CF0100'         STANDARD SUFFIX DATA
         LA    R0,5                       FIXED LENGTH
         STH   R0,0(R2)                    OF FIVE
         LA    R2,5(R2)                   INCREMENT BUFFER POINTER
         BR    R14                         & RETURN
         SPACE 2
************************************************************************
*  COMMANDS                                                            *
************************************************************************
         SPACE
FILECMD  DS    0F              BUILD A DIU 'FILE' COMMAND
*                                         LL ADDED LATER
         MVC   2(3,R2),=X'CC0201'         MOVE CTF
         LA    R2,5(R2)                   INCR BUFFER POINTER
         BR    R14                        RETURN TO CALLER
         SPACE 2
SONCMD   DS    0F              BUILD A DIU 'SIGNON' COMMAND
*                                         LL ADDED LATER
         MVC   2(3,R2),=X'CD0C01'         MOVE CTF
         LA    R2,5(R2)                   INCR BUFFER POINTER
         BR    R14                        RETURN TO CALLER
         SPACE 2
OBTCMD   DS    0F              BUILD A DIU 'OBTAIN' COMMAND
*                                         LL ADDED LATER
         MVC   2(3,R2),=X'CC1701'         MOVE CTF
         LA    R2,5(R2)                   INCR BUFFER POINTER
         BR    R14                        RETURN TO CALLER
         SPACE 2
DISCMD   DS    0F              BUILD A DIU 'DISTRIBUTE' COMMAND
*                                         LL ADDED LATER
         MVC   2(3,R2),=X'CC1C01'         MOVE CTF
         LA    R2,5(R2)                   INCR BUFFER POINTER
         BR    R14                        RETURN TO CALLER
         SPACE 2
************************************************************************
*  PARAMETERS FOR 'FILE' & 'DISTRIBUTE'                                *
************************************************************************
         SPACE
IDDATA1  DS    0F
         MVC   2(3,R2),=X'C52001'         MOVE CTF
         STC   R0,5(R2)                   IDENTIFIED DATA IN R0
         LA    R0,6
         STH   R0,0(R2)                   SET LL
         LA    R2,6(R2)                   INCR BUFFER POINTER
         BR    R14
         SPACE 2
SRCADDR  DS    0F                   SOURCE ADDRESS
         LR    R15,R0                     LENGTH OF NAME
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C32301'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCSRCAD               MOVE IN NAME
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCSRCAD MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
ACCCODE  DS    0F
         MVC   2(3,R2),=X'C33941'         SET CTF VALUE
         MVC   5(2,R2),=X'0601'           LENGTH + TYPE
         L     R0,0(R1)                   LOAD ACCESS CODE VALUE
         ST    R0,7(R2)                   SET ACCESS CODE VALUE
         LA    R0,11
         STH   R0,0(R2)                   SET LL VALUE
         LA    R2,11(R2)                  UPDATE TOTAL LENGTH
         BR    R14
         SPACE 2
DUSEGN   DS    0F
*                                         LL SET LATER
         MVC   2(6,R2),=X'C90381200000'   SEGMENT INTRODUCER
         LA    R2,8(R2)                   INCREMENT BUFFER PTR
         BR    R14
         SPACE 2
DUSEGL   DS    0F
*                                         LL SET LATER
         MVC   2(6,R2),=X'C90381000000'   SEGMENT INTRODUCER
         LA    R2,8(R2)                   INCREMENT BUFFER PTR
         BR    R14
         SPACE 2
DUSYS    DS    0F
*                                         LL SET LATER
         STH   R0,0(R2)                   SET DOC TYPE
         MVC   2(13,R2),0(R1)             MOVE SYSTEM ID
         LA    R2,15(R2)                  INCR BUF PTR
         BR    R14
         SPACE 2
PRDOC    DS    0F
*                                         LL SET LATER
         MVC   2(3,R2),=X'CA0301'         CTF
         LA    R2,5(R2)                   INCR BUF PTR
         BR    R14
         SPACE 2
PRBASE   DS    0F
*                                         LL SET LATER
         MVC   2(3,R2),=X'CA0401'         CTF
         LA    R2,5(R2)                   INCR BUF PTR
         BR    R14
         SPACE 2
PRBTYPE  DS    0F
         MVC   2(3,R2),=X'C70601'         CTF
         STH   R0,5(R2)                   DOC TYPE
         LA    R0,7
         STH   R0,0(R2)                   SET LL VALUE
         LA    R2,7(R2)                   INCR BUF PTR
         BR    R14
         SPACE 2
PRBGCID  DS    0F
         MVC   2(3,R2),=X'C70101'         CTF
         ST    R0,5(R2)                   SET GCID/GPID
         LA    R0,9
         STH   R0,0(R2)                   SET LL VALUE
         LA    R2,9(R2)                   INCR BUF PTR
         BR    R14
         SPACE 2
DGCID    DS    0F
         MVC   2(7,R2),=X'C7050100D70108' GCID 215-264 '1403'
*        MVC   2(7,R2),=X'C7050101510100' GCID 337-256 'FOR TESTING'
         LA    R0,9
         STH   R0,0(R2)                   SET LL
         LA    R2,9(R2)                   INCR BUFFER POINTER
         BR    R14
         SPACE 2
PRBDOCN  DS    0F
         LR    R15,R0                     LENGTH OF DOC NAME
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C70001'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCPRBD                MOVE IN DOC NAME
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCPRBD  MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
PRBSUBJ  DS    0F
         LR    R15,R0                     LENGTH OF SUBJECT FIELD
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C70B01'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCPRBS                MOVE IN SUBJECT
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCPRBS  MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
PRBAUTH  DS    0F
         LR    R15,R0                     LENGTH OF AUTHOR FIELD
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C70401'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCPRBA                MOVE IN AUTHOR NAME
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCPRBA  MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
PRBTIME  DS    0F
         MVC   2(3,R2),=X'C70701'         CTF VALUE
         MVC   5(6,R2),DATETIME           STD VALUE FOR THE MOMENT
         LA    R0,11                      LENGTH
         STH   R0,0(R2)
         LA    R2,11(R2)                  POINT TO NEXT FREE BYTE
         BR    R14
         SPACE
DATETIME DC    AL2(1983),AL1(05),AL1(21),AL1(22),AL1(30)
* 22.30 ON 21ST MAY 1983
         SPACE
***********************************************************************
*  PARAMETERS FOR 'SIGNON'                                            *
***********************************************************************
         SPACE
FUNCSETS DS    0F
         LR    R15,R0                     LENGTH OF NAME FIELD
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C31201'         CTF VALUE
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCFUNC                MOVE IN FUNC SETS STRING
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCFUNC  MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
SONNAME  DS    0F
         LR    R15,R0                     LENGTH OF NAME FIELD
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C30D01'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCSONN                MOVE IN USER NAME
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCSONN  MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
SONPASS  DS    0F
```

```
         LR    R15,R0                     LENGTH OF PASSWORD
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C33801'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCSONP                MOVE IN PASSWORD
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCSONP  MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
RCVDOCS  DS    0F                        SINGLE RECEIVE DOC TYPE
         MVC   2(3,R2),=X'C32901'         CTF
         STH   R0,5(R2)                   STORE DOC TYPE
         LA    R0,7                       LENGTH
         STH   R0,0(R2)                   STORED IN LL FIELD
         LA    R2,7(R2)                   BUMP BUFPTR
         BR    R14
         SPACE
***********************************************************************
*  PARAMETERS FOR 'OBTAIN'                                            *
***********************************************************************
         SPACE
OBTOPTS  DS    0F                         OBTAIN-OPTIONS
         MVC   2(3,R2),=X'C31E01'         CTF
         STC   R0,5(R2)                   STORE OPTION BYTE
         LA    R0,6                       LENGTH
         STH   R0,0(R2)                   STORED IN LL FIELD
         LA    R2,6(R2)                   BUMP BUFPTR
         BR    R14
         SPACE
***********************************************************************
*  PARAMETERS FOR 'DISTRIBUTE'                                        *
***********************************************************************
         SPACE
DNADDR   DS    0F                   DESTINATION NODE ADDRESS
         LR    R15,R0                     LENGTH OF ADDRESS NAME
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C32F01'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCDNAD                MOVE IN ADDRESS
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCDNAD  MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
ATTLST   DS    0F
         MVC   2(7,R2),=X'C3050100000001' MOVE CTFATL
         LA    R0,9
         STH   R0,0(R2)                   SET LL
         LA    R2,9(R2)                   INCR BUFFER POINTER
         BR    R14
         SPACE 2
RECADDR  DS    0F                   RECIPIENT NAME
         LR    R15,R0                     LENGTH OF NAME
         LA    R0,5(R15)                  ADD 5 FOR LLCTF
         STH   R0,0(R2)                   STORE LL VALUE
         MVC   2(3,R2),=X'C30601'         MOVE CTF DATA
         BCTR  R15,0                      -1 FOR EXECUTE
         EX    R15,MVCRECAD               MOVE IN NAME
         LA    R2,6(R15,R2)               POINT AT NEXT AVAIL BYTE
         BR    R14                        & RETURN
         SPACE
MVCRECAD MVC   5(1,R2),0(R1)              EXECUTED MOVE
         SPACE
***********************************************************************
*  DOCUMENT BUILDERS                                                  *
***********************************************************************
         SPACE
DCI      DS    0F
         MVC   2(3,R2),=X'CB0101'
         LA    R0,5
         STH   R0,0(R2)
         LA    R2,5(R2)
         BR    R14
```

B.4.4 APIDISOS

```
DIS  TITLE '*APIDISOS* - CALL DISOSS INTERFACE - DISOSS API SAMPLE PROG*
               RAMS'
         PRINT NOGEN
APIDISOS CSECT
***********************************************************************
*                                                                     *
*   (APIDISOS) - SUBROUTINE TO DO DISOSS CALLS                        *
*                                                                     *
*   INPUT PARMS:   -> APICOM                                          *
*                  -> COMMAND VERB                                    *
*   ON EXIT, THE DISOSS RETURN CODE IS IN COMRETCD                    *
*                                                                     *
***********************************************************************
         SPACE
         L     R8,4(R1)              LOAD APICOM ADDRESS
         USING APICOM,R8
         L     R4,8(R1)              LOAD ADDRESS OF VERB
         SPACE
         MVC   COMXN,=C'*APICOM*'    & EYE CATCHER
         MVC   COMXC,0(R4)           COMMAND VERB
         SPACE
         LA    R0,COMXC              ADDRESS THE COMMAND VERB
         ST    R0,COMP1              STORE ADDRESS OF COMMAND VERB
         XC    COMRETCD,COMRETCD     ZERO RETCODE FIELD
         LA    R0,COMRETCD           RETCODE FULLWORD
         ST    R0,COMP2
         LA    R0,COMCPTR            CONTROL BLOCK POINTER
         ST    R0,COMP3
         LA    R0,COMDPTR            DATA POINTER
         ST    R0,COMP4
         LA    R0,COMDLEN            DATA LENGTH
         ST    R0,COMP5
         OI    COMP5,X'80'           SET VL BIT
         SPACE
* TRACE IF TRACE FLAG IS SET
         CLI   COMTRFLG,COMTRYES     SHALL WE TRACE ?
         BNE   DIS1                  NO
         STM   R0,R15,COMTRREG       SAVE ALL REGS
         EXEC CICS ENTER TRACEID(1)
         LM    R0,R15,COMTRREG       RELOAD REGS
DIS1     EQU   *
         SPACE
* NOW ASK DISOSS TO PROCESS THE REQUEST
         LA    R1,COMPARMS           R1 -> PARM LIST
         L     R15,=V(DSVAW000)      INTERFACE ROUTINE ADDRESS
         BALR  R14,R15               CALL DISOSS
         SPACE
* TRACE IF TRACE FLAG IS SET
         CLI   COMTRFLG,COMTRYES     SHALL WE TRACE ?
         BNE   DIS2                  NO
         STM   R0,R15,COMTRREG       SAVE ALL REGS
         EXEC CICS ENTER TRACEID(2)
         LM    R0,R15,COMTRREG       RELOAD REGS
DIS2     EQU   *
         EJECT
DFHEISTG DSECT
         PRINT OFF
```

```
COPY  APICOM
COPY  APIREGS
PRINT ON
END
```

B.4.5 APIDIS2

This is a modification of the original APIDISTR.

```
DIST TITLE '*APIDIS2* - BUILD AND SEND A DIA DISTRIBUTE COMMAND -  DISO*
               SS API SAMPLE PROGRAMS'
        PRINT NOGEN
APIDIS2 CSECT
***********************************************************************
* (APIDIS2) - BUILDS A DIA 'DISTRIBUTE' COMMAND AND   PROFILE INFO    *
*          THEN SENDS IT TO DISOSS                                    *
* THIS IS A COPY OF APIDISTR, ADDED FUNCTIONS ARE:                    *
*                -EXPANDED DIU CORRELATION TO 16 BYTES                *
*   INPUT PARM LIST:                                                  *
*             -> APIDPR (DOCUMENT PROFILE MAP)                        *
*             -> DIU BUFFER (4K)                                      *
*   OUTPUT: RETURN CODE IN COMRETCD                                   *
*              0  COMMAND BUILT AND SENT OK                           *
*              4  ERROR IN DPR PARAMETERS                             *
*              8  BAD RETURN FROM DISOSS. VALUE IN COMREASN           *
***********************************************************************
        SPACE
        L      R8,4(R1)            GET COM ADDRESS
        USING  APICOM,R8
        L      R9,8(R1)            ADDRESS OF DPR MAP
        USING  APIDPR,R9
        L      R10,12(R1)          ADDRESS OF DIU BUFFER
        LR     R2,R10              FOR DIU BUILDERS
        EJECT
***********************************************************************
* HERE WE BUILD THE DIU BASED ON THE DPR INFORMATION                  *
***********************************************************************
        SPACE
* DIU PREFIX
        LA     R1,DPRCOR           SET CORRELATION DATA
        LA     R0,16                CORRELATION LENGTH ALWAYS 16
        BAL    R14,DIUPFX          BUILD DIU PREFIX
* COMMAND SEQUENCE - 'DISTRIBUTE'
        LR     R4,R2               SAVE ADDR OF CMD SEQUENCE LL
        BAL    R14,DISCMD         BUILD 'DISTRIBUTE' COMMAND
* IDENTIFIED DATA: DOCUMENT IS IN FIRST DOCUMENT UNIT
        LA     R0,1                = 1ST DOC UNIT
        BAL    R14,IDDATA1         IDENTIFIED DATA (FORMAT 1)
* DESTINATION NODE ADDRESS
        LA     R0,8                ALWAYS LENGTH 8
        LA     R1,DPRDDN           DESTINATION NODE
        BAL    R14,DNADDR
* ATTRIBUTE LIST, NO COD, NOT PERSONAL, NO PRIORITY, COPIES = 1
        BAL    R14,ATTLST          ATTRIBUTE LIST
* RECIPIENT NAME
        LA     R0,8                ALWAYS LENGTH 8
        LA     R1,DPRRID           RECIPIENT NAME
        BAL    R14,RECADDR
* SET LENGTH OF COMMAND SEQUENCE SEGMENT
        LR     R0,R2               SET CURRENT POINTER
        LR     R1,R4               COLLECT ADDR OF CMD SEQ LL
        SLR    R0,R1               SUBTRACT ADDR OF LL
        STH    R0,0(R1)           SET CMD SEQUENCE LL
* NOW START DOCUMENT UNIT SEGMENTS
        LR     R4,R2              USE R4 FOR ADDR OF LL
        BAL    R14,DUSEGN         DOCUMENT UNIT SEGMENT (NOT LAST)
* SET DOC TYPE AND SYSTEM ID
        LH     R0,DPRDOT          SET DOC TYPE AS PASSED
        LA     R1,DPRSYS           AND SYSTEM ID (IGNORE LENGTHS)
        BAL    R14,DUSYS          SET DOC TYPE & SYSTEM ID
* START OF DOCUMENT PROFILE(S)
        LR     R5,R2              ADDR OF PROFILE LL
        BAL    R14,PRDOC          BUILD DOC PROFILE LLCTF
* BASE SUBPROFILE
        LR     R6,R2              ADDR OF BASE SUBPROFILE LL
        BAL    R14,PRBASE         BUILD BASE SUBPROFILE LLCTF
* DOCUMENT TYPE
        LH     R0,DPRDOT          DOCTYPE AS PASSED
        BAL    R14,PRBTYPE        SET DOCUMENT TYPE
* PROFILE GCID
        L      R0,DPRPGC          GPID + GCID
        BAL    R14,PRBGCID        SET PROFILE GCID
* DOCUMENT GCID
* DISOSS DOES NOT LOOK AT GCID IN PROFILE
*       BAL    R14,DGCID          DOCUMENT GCID 215-108
* DOCUMENT NAME
        LA     R1,DPRDON          DOCUMENT NAME
        SLR    R0,R0
        IC     R0,DPRDONL         & LENGTH
        BAL    R14,PRBDOCN        SET DOCUMENT NAME
* SUBJECT
        LA     R1,DPRSUB
        SLR    R0,R0
        IC     R0,DPRSUBL
        BAL    R14,PRBSUBJ        SET SUBJECT FIELD
* AUTHOR
        LA     R1,DPRAUT
        SLR    R0,R0
        IC     R0,DPRAUTL
        BAL    R14,PRBAUTH        SET AUTHOR VALUE
* CREATION DATE & TIME
*       BAL    R14,PRBTIME        SET CURRENT DATE AND TIME
* END OF BASE SUBPROFILE  - SET BASE SUBPROFILE LL
        LR     R0,R2              GET CURRENT ADDRESS
        SLR    R0,R6              GET LENGTH OF BASE SUBPROF
        STH    R0,0(R6)           & STORE LENGTH IN LL FIELD
* END OF DOCUMENT PROFILES  - SET PROFILE LL
        LR     R0,R2              GET CURRENT ADDRESS
        SLR    R0,R5              GET LENGTH OF DOC PROFILE
        STH    R0,0(R5)           & STORE LENGTH IN LL FIELD
* DOCUMENT CONTENT INTRODUCER
        BAL    R14,DCI            DCI BUILDER
* END OF DOC UNIT SEGMENT - SET DOC UNIT LL
        LR     R0,R2              GET CURRENT ADDRESS
        SLR    R0,R4              GET LENGTH OF DOC UNIT
        STH    R0,0(R4)           & STORE LENGTH IN LL FIELD
        EJECT
**********************************************************************
* DIU NOW BUILT - SEND IT TO DISOSS                                  *
**********************************************************************
        SPACE
* TELL DISOSS WHAT WE HAVE
```

```
        ST     R10,COMDPTR        TELL DISOSS WHERE IT IS
        SLR    R2,R10             OBTAIN LENGTH
        STH    R2,COMDLEN         TELL DISOSS ALSO
        SPACE
        APICALL APIDISOS,(=CL8'SEND')
        SPACE
* WE RETURN WITH THE RETURN CODE FROM DISOSS
         ST    R15,COMREASN       SET REASON CODE AS DISOSS RC
         LTR   R15,R15           WAS IT BAD RC ?
         BZ    *+8                NO, USE ZERO AS OUR RC
         LA    R15,8              OTHERWISE WE HAVE RC 8
         ST    R15,COMRETCD
         SPACE
         B     ENDCSECT           BRANCH ROUND STATIC DATA ETC
         EJECT
         COPY  APIDIUS2           DIU BUILDER SUBROUTINES
         EJECT
ENDCSECT DS    0H
DFHEISTG DSECT
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR2
         COPY  APIREGS
         PRINT ON
         END
```

B.4.6 APIFIL2

This is a modification of the original APIFILE.

```
FILE TITLE '*APIFILE* - BUILD AND SEND A DIA FILE COMMAND - DISOSS API *
               SAMPLE PROGRAMS'
        PRINT NOGEN
APIFIL2 CSECT
*********************************************************************
* (FILE) - BUILDS A 'FILE' COMMAND, PLUS PROFILE INFO               *
*          THEN SENDS IT TO DISOSS                                  *
* COPY OF APIFILE, ADDED                                            *
*          16 BYTES CORR. DATA                                      *
*          SOURCE ADDRESS, FILE ON BEHALF..                         *
*   INPUT PARM LIST:                                                *
*             -> APIDPR (DOCUMENT PROFILE MAP)                      *
*             -> DIU BUFFER (4K)                                    *
*   OUTPUT: RETURN CODE IN COMRETCD                                 *
*              0  COMMAND BUILT AND SENT OK                         *
*              4  ERROR IN DPR PARAMETERS                           *
*              8  BAD RETURN FROM DISOSS. VALUE IN COMREASN         *
*********************************************************************
        SPACE
        L      R8,4(R1)           GET COM ADDRESS
        USING  APICOM,R8
        L      R9,8(R1)           ADDRESS OF DPR MAP
        USING  APIDPR,R9
        L      R10,12(R1)         ADDRESS OF DIU BUFFER
        LR     R2,R10             FOR DIU BUILDERS
        EJECT
*********************************************************************
* HERE WE BUILD THE DIU BASED ON THE DPR INFORMATION                *
*********************************************************************
        SPACE
* DIU PREFIX
*       LA     R1,=C'03'          SET ADDRESS AND
*       LA     R0,2                LENGTH OF CORRELATION DATA
        LA     R1,DPRCOR           CORRELATION DATA
        LA     R0,16               LENGTH OF CORRELATION DATA
        BAL    R14,DIUPFX         BUILD DIU PREFIX
* COMMAND SEQUENCE - 'FILE'
        LR     R4,R2              SAVE ADDR OF CMD SEQUENCE LL
        BAL    R14,FILECMD        BUILD 'FILE' COMMAND
* IDENTIFIED DATA: DOCUMENT IS IN FIRST DOCUMENT UNIT
        LA     R0,1               = 1ST DOC UNIT
        BAL    R14,IDDATA1        IDENTIFIED DATA (FORMAT 1)
* SOURCE ADDRESS FILE ON BEHALF OF
        LA     R0,8               SET LENGTH ALWAYS 8
        LA     R1,DPRRID          RECIPIENT NAME USED AS SOURCE ADRESS
        BAL    R14,SRCADDR
* ACCESS CODE VALUE
        SLR    R0,R0              CLEAR REG 0
        IC     R0,DPRACCL         SET LENGTH
        LA     R1,DPRACC          SET ACC CODE VALUE
        BAL    R14,ACCCODE
* SET LENGTH OF COMMAND SEQUENCE SEGMENT
        LR     R0,R2              SET CURRENT POINTER
        LR     R1,R4              COLLECT ADDR OF CMD SEQ LL
        SLR    R0,R1              SUBTRACT ADDR OF LL
        STH    R0,0(R1)           SET CMD SEQUENCE LL
* NOW START DOCUMENT UNIT SEGMENTS
        LR     R4,R2              USE R4 FOR ADDR OF LL
        BAL    R14,DUSEGN         DOCUMENT UNIT SEGMENT (NOT LAST)
* SET DOC TYPE AND SYSTEM ID
        LH     R0,DPRDOT          SET DOC TYPE AS PASSED
        LA     R1,DPRSYS           AND SYSTEM ID (IGNORE LENGTHS)
        BAL    R14,DUSYS          SET DOC TYPE & SYSTEM ID
* START OF DOCUMENT PROFILE(S)
        LR     R5,R2              ADDR OF PROFILE LL
        BAL    R14,PRDOC          BUILD DOC PROFILE LLCTF
* BASE SUBPROFILE
        LR     R6,R2              ADDR OF BASE SUBPROFILE LL
        BAL    R14,PRBASE         BUILD BASE SUBPROFILE LLCTF
* DOCUMENT TYPE
        LH     R0,DPRDOT          DOCTYPE AS PASSED
        BAL    R14,PRBTYPE        SET DOCUMENT TYPE
* PROFILE GCID
        L      R0,DPRPGC          GPID + GCID
        BAL    R14,PRBGCID        SET PROFILE GCID '01510100'
* DOCUMENT GCID
* DISOSS DOES NOT LOOK INTO DOC GCID IN PROFILE
*       BAL    R14,DGCID          SET DOCUMENT GCID '00D70108'
* DOCUMENT NAME
        LA     R1,DPRDON          DOCUMENT NAME
        SLR    R0,R0
        IC     R0,DPRDONL         & LENGTH
        BAL    R14,PRBDOCN        SET DOCUMENT NAME
* SUBJECT
        LA     R1,DPRSUB
        SLR    R0,R0
        IC     R0,DPRSUBL
        BAL    R14,PRBSUBJ        SET SUBJECT FIELD
* AUTHOR
        LA     R1,DPRAUT
        SLR    R0,R0
        IC     R0,DPRAUTL
        BAL    R14,PRBAUTH        SET AUTHOR VALUE
* CREATION DATE & TIME
*       BAL    R14,PRBTIME        SET CURRENT DATE AND TIME
* END OF BASE SUBPROFILE  - SET BASE SUBPROFILE LL
        LR     R0,R2              GET CURRENT ADDRESS
        SLR    R0,R6              GET LENGTH OF BASE SUBPROF
        STH    R0,0(R6)           & STORE LENGTH IN LL FIELD
* END OF DOCUMENT PROFILES  - SET PROFILE LL
        LR     R0,R2              GET CURRENT ADDRESS
        SLR    R0,R5              GET LENGTH OF DOC PROFILE
        STH    R0,0(R5)           & STORE LENGTH IN LL FIELD
* DOCUMENT CONTENT INTRODUCER
        BAL    R14,DCI            DCI BUILDER
* END OF DOC UNIT SEGMENT - SET DOC UNIT LL
        LR     R0,R2              GET CURRENT ADDRESS
        SLR    R0,R4              GET LENGTH OF DOC UNIT
        STH    R0,0(R4)           & STORE LENGTH IN LL FIELD
        EJECT
*********************************************************************
* DIU NOW BUILT - SEND IT TO DISOSS                                 *
*********************************************************************
```

```
        SPACE
* TELL DISOSS WHAT WE HAVE
        ST     R10,COMDPTR        TELL DISOSS WHERE IT IS
        SLR    R2,R10             OBTAIN LENGTH
        STH    R2,COMDLEN         TELL DISOSS ALSO
        SPACE
        APICALL APIDISOS,(=CL8'SEND')
        SPACE
* WE RETURN WITH THE RETURN CODE FROM DISOSS
         ST    R15,COMREASN       SET REASON CODE AS DISOSS RC
         LTR   R15,R15           WAS IT BAD RC ?
         BZ    *+8                NO, USE ZERO AS OUR RC
         LA    R15,8              OTHERWISE WE HAVE RC 8
         ST    R15,COMRETCD
         SPACE
         B     ENDCSECT           BRANCH ROUND STATIC DATA ETC
         EJECT
         COPY  APIDIUS2           DIU BUILDER SUBROUTINES
         EJECT
ENDCSECT DS    0H
DFHEISTG DSECT
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR2
         COPY  APIREGS
         PRINT ON
         END
```

B.4.7 APIGTCMD

```
GCMD TITLE '*APIGTCMD* - GET COMMAND && PARMS FROM INPUT DIU - DISOSS A*
               PI SAMPLE PROGRAMS'
         PRINT NOGEN
APIGTCMD CSECT
***********************************************************************
*                                                                     *
* (APIGTCMD)  ANALYZES DIU INPUT AND INDICATES COMMAND IN THE         *
*             COM BLOCK AND THE PARAMETERS IN THE PASSED DPR BLOCK    *
*                                                                     *
***********************************************************************
         SPACE
         L     R8,4(R1)            ADDR OF APICOM
         USING APICOM,R8
         L     R4,8(R1)            ADDRESS OF DPR BLOCK
         USING APIDPR,R4
         SPACE
GETCMD1  EQU   *
* WE PASS ON OUR PASSED 4K DPR BLOCK
         APICALL APIPARSE,(APIDPR)
         SPACE
         LTR   R15,R15             MAKE SURE OK.
         BNZ   GETCMDR             RETURN IF NOT
         TM    COMDST,COMDSTC      HAVE WE GOT COMMAND YET ?
         BZ    GETCMD1             NO, TRY FOR NEXT
GETCMDR  EQU   *
         EJECT
DFHEISTG DSECT
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR
         COPY  APIREGS
         PRINT ON
         END
```

B.4.8 APILAST

```
LAST TITLE '*APILAST* - PERFORM A "LAST" CALL - DISOSS API SAMPLE PROGR*
               AMS'
         PRINT NOGEN
APILAST  CSECT
***********************************************************************
*                                                                     *
*   (LAST)  SUBROUTINE TO ISSUE A 'LAST' CALL TO DISOSS               *
*      PARMS:                                                         *
*        NAME OF TRANSACTION THAT DISOSS SHOULD START IN RESPONSE     *
*                                                                     *
***********************************************************************
         SPACE
         L     R8,4(R1)            LOAD APICOM ADDRESS
         USING APICOM,R8
         L     R4,8(R1)            ADDR OF TRANSACTION NAME
         SPACE
         MVC   LSTTRAN,0(R4)       RESPONSE TRANSACTION ID
         MVI   LSTFLAG,X'FF'       1 RESPONSE TRAN FOR ALL
         MVC   LSTTERM,EIBTRMID    RESPONSE TRAN FOR THIS TERMINAL
         SPACE
         LA    R0,LST              GET COMMAND DATA
         ST    R0,COMDPTR           & TELL DISOSS
         LA    R0,LSTLEN           GET DATA LENGTH
         STH   R0,COMDLEN           TELL DISOSS
         SPACE
* NOW CALL DISOSS
         APICALL APIDISOS,(=CL8'LAST')
         SPACE
         ST    R15,COMREASN       SET REASON CODE AS DISOSS RC
         LTR   R15,R15            WAS IT BAD RC ?
         BZ    *+8                NO, USE ZERO AS OUR RC
         LA    R15,8              OTHERWISE WE HAVE RC 8
         ST    R15,COMRETCD
         EJECT
DFHEISTG DSECT
LST      DS    0F                     PARAMETERS FOR "LAST"
LSTTRAN  DS    CL4                    RESPONSE TRANSACTION ID
LSTFLAG  DS    X                      1 TRANSACTION PER MSG
LSTTERM  DS    CL4                    TERMID TO START TRAN AGAINST
LSTLEN   EQU   *-LST
         SPACE
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR
         COPY  APIREGS
         PRINT ON
         END
```

B.4.9 APIPARSE

```
PARS    TITLE '*APIPARSE* - DIU PARSER - DISOSS API SAMPLE PROGRAMS'
         PRINT NOGEN
APIPARSE CSECT
***********************************************************************
*                                                                     *
*   (APIPARSE) PARSE AN INCOMIMG DIU ELEMENT                          *
*           RECEIVE THE NEXT DIU SEGMENT FROM DISOSS IF NECESSARY.    *
*                                                                     *
*    INPUT:  ->  4K DPR OR TEXT BUFFER TO RECEIVE PARAMETERS OR       *
*                PROFILE OR DOCUMENT SEGMENT.                         *
*                                                                     *
***********************************************************************
         SPACE
         L     R8,4(R1)            APICOM ADDRESS
         USING APICOM,R8
         L     R2,8(R1)            HOLD 4K BLOCK POINTER IN R2 HERE
         USING APIDPR,R2
         SPACE
         L     R10,COMDBUFP        LOAD CURRENT DIU BUFFER ADDRESS
         LTR   R10,R10             DO WE HAVE ONE ?
         BNZ   ANDIU0              YES, SKIP GETMAIN
         EXEC CICS GETMAIN SET(R10) LENGTH(8220) INITIMG(X'00')
         ST    R10,COMDBUFP        STORE ITS ADDRESS
         SLR   R0,R0               INITIALIZE ...
         STH   R0,COMLDIU1         - TOTAL BUFFER LENGTH
         STH   R0,COMLDIU2         - TOTAL DIU SEGMENT LENGTH
         STH   R0,COMLDIU3         - AMOUNT OF SEGMENT USED SO FAR
         MVI   COMDCMD,X'00'       NO COMMAND AS YET
         MVI   COMDST,X'00'        STATUS INDETERMINATE
         SPACE
ANDIU0   EQU   *
         TM    COMDST,COMDSTE      HAVE WE REACHED END OF DATA ?
         BO    ANDIU2              YES, NO MORE TO GET
         LH    R0,COMLDIU1         HOW MUCH IN BUFFER ?
         CH    R0,=H'4110'         IF .GT 4110
         BH    ANDIU2              DON'T GET MORE
         APICALL APIRECVE          OTHERWISE GET MORE FROM DISOSS
         LTR   R15,R15             CHECK RC
         BZ    ANDIU1              0 - CARRY ON
         CH    R0,=H'3'            WAS IT END OF DATA ?
         BNE   ANDIUR              NO, QUIT WITH BAD RETURN
         XC    COMRETCD,COMRETCD   CLEAR BAD RETURN CODE INDICATOR
         OI    COMDST,COMDSTE      SET END OF DATA FLAG
         B     ANDIU2              AND GO PROCESS.
         SPACE
* WE HAVE SOME DATA. MOVE IT INTO OUR BUFFER
ANDIU1   EQU   *
         LR    R4,R10              GET BUFFER ADDRESS
         AH    R4,COMLDIU1         INCR TO CURRENT POSITION
         L     R6,COMDPTR          ADDRESS OF INPUT DATA
         LH    R7,COMLEN           SOURCE LENGTH
         LR    R5,R7               TARGET LENGTH = SOURCE
         MVCL  R4,R6               MOVE INPUT DATA
* AND MAINTAIN RECORD OF THIS DATA
         LH    R7,COMLDIU1         CURRENT LENGTH
         AH    R7,COMLEN           PLUS NEW LENGTH
         STH   R7,COMLDIU1         UPDATED
         B     ANDIU0              GET NEXT AS NECESSARY
         SPACE
ANDIU2   EQU   *                   DIU ANALYSIS
         LA    R4,ANTAB            POINT AT C-T TABLE
         LA    R6,8                LENGTH OF TABLE ENTRY
         LA    R7,ANTABE-1         POINT AT TABLE END
ANDIU3   L     R5,4(R4)            GET BRANCH ADDRESS
         CLC   2(2,R10),0(R4)      IS IT THIS TABLE ENTRY ?
         BER   R5                  YES, BRANCH TO PROCESSING ROUTINE
         BXLE  R4,R6,ANDIU3        OTHERWISE LOOP FOR NEXT
* DIU C-T BYTE NOT RECOGNIZED - ERROR
         LA    R15,255             INDICATE MAJOR ERROR
         B     ANDIUR              AND RETURN TO CALLER
         SPACE 2
ANCMDCH  DS    0F
         LR    R4,R9                 COPY LL VALUE FOR ELEMENT
         AH    R4,COMLDIU3           ADD IN AMOUNT USED
         CH    R4,COMLDIU2           HAVE WE PROCESSED ALL SEGMENT
         BL    ANEND                 NO, RETURN
         OI    COMDST,COMDSTC        YES, MARK COMMAND COMPLETE
         B     ANEND                 & RETURN
         SPACE 2
ANPRFCH  DS    0F
         LR    R4,R9                 COPY LL VALUE FOR ELEMENT
         AH    R4,COMLDIU3           ADD IN AMOUNT USED
         CH    R4,COMLDIU2           HAVE WE PROCESSED ALL SEGMENT
         BL    ANEND                 NO, RETURN
         OI    COMDST,COMDSTP        YES, MARK PROFILES COMPLETE
         B     ANEND                 & RETURN
         SPACE 2
ANEND    EQU   *
* WE SHIFT ALL THE BUFFER LEFT OVER THE CURRENT ELEMENT
         LR    R4,R10              TARGET ADDRESS
         LA    R6,0(R9,R4)         SOURCE ADDRESS
         LH    R7,COMLDIU1         CURRENT LENGTH
         SLR   R7,R9               LESS LL FOR CURRENT ELEMENT
         LR    R5,R7               TARGET LENGTH
         MVCL  R4,R6               SHIFT LEFT IN BUFFER
         SPACE
* UPDATE LENGTH POINTERS
         LH    R7,COMLDIU1         TOTAL LENGTH
         SLR   R7,R9               LESS AMOUNT JUST PROCESSED
         STH   R7,COMLDIU1         UPDATED
         LH    R7,COMLDIU3         AMOUNT USED IN SEGMENT
         ALR   R7,R9               + ELEMENT JUST PROCESSED
         STH   R7,COMLDIU3         UPDATED
         SPACE
         TM    COMDST,COMDSTE      DID WE HIT END OF DATA ?
         BZ    ANDIUR1             NO
         LH    R0,COMLDIU1         YES, SO GET DATA LENGTH
         LTR   R0,R0               ANYTHING LEFT ?
         BNZ   ANDIUR1             YES
         L     R10,COMDBUFP        NO, LOAD ADDR OF BUFFER
         EXEC CICS FREEMAIN DATA(0(R10))
         XC    COMDBUFP,COMDBUFP   CLEAR ADDRESS
         SPACE
ANDIUR1  EQU   *
         SLR   R15,R15             INDICATE GOOD RETURN
         SPACE
ANDIUR   EQU   *
```

```
         STM   R0,R15,COMTRREG
         EXEC CICS ENTER TRACEID(7)
         LM    R0,R15,COMTRREG
         ST    R0,COMREASN         SET OUR REASON CODE
         ST    R15,COMRETCD        AND OUR CALCULATED RC
         B     ENDCSECT            FINISHED
         EJECT
***********************************************************************
*                                                                     *
*   (ANXX)    DIU ANALYSIS ROUTINES, REACHED BY BRANCH TABLE          *
*                                                                     *
***********************************************************************
         SPACE
* PREFIX
AN10     EQU   *
         MVI   COMDCMD,X'00'         CLEAR COMMAND BYTE
         NI    COMDST,255-(COMDSTC+COMDSTP+COMDSTD+COMDSTX)
         LH    R9,0(R10)             GET LL VALUE
         B     ANEND                 BUMP AND CHECK LENGTH
* SUFFIX
AN11     EQU   *
         OI    COMDST,COMDSTX        SHOW SUFFIX FOUND
         LH    R9,0(R10)             SO WE CLEANLY HAVE ZERO LENGTH
         B     ANEND                 GO SCHEDULE DISOSS
         SPACE 2
* DELIVER COMMAND
AN15     EQU   *
         MVI   COMDCMD,COMDDLV     INDICATE THIS IS DELIVER
         B     ANSTCMD              SETUP FOR START OF COMMAND
* ACKNOWLEDGE COMMAND
AN16     EQU   *
         MVI   COMDCMD,COMDACK      INDICATE THIS IS ACK
         B     ANSTCMD              SETUP FOR START OF COMMAND
* SIGNON RESPONSE
AN17     EQU   *
         MVI   COMDCMD,COMDSON      INDICATE THIS IS SIGNON RESPONSE
         B     ANSTCMD              SETUP FOR START OF COMMAND
         SPACE 2
ANSTCMD  EQU   *   START OF COMMAND SEGMENT
         SLR   R0,R0
         STH   R0,COMLDIU3           NOTHING USED IN SEGMENT SO FAR
         LH    R9,0(R10)             GET COMMAND UNIT LENGTH
         STH   R9,COMLDIU2           SO WE KNOW WHEN END OF COMMAND
         LA    R9,5                  JUST 5 FOR COMMAND
         B     ANEND                 BUMP AND CHECK LENGTH
         EJECT
* THE FOLLOWING COMMAND OPERANDS WE IGNORE, EXCEPT THAT WE CHECK
* TO SEE IF THE COMMAND OPERANDS ARE COMPLETE
         SPACE
AN20     EQU   *                     IDENTIFIED DATA
AN21     EQU   *                     CORRELATION
AN24     EQU   *                     ATTRIBUTE LIST
AN25     EQU   *                     RECIPIENT NAME (WE KNOW IT)
AN40     EQU   *                     FUNCTION SETS
AN41     EQU   *                     SIGNON REPLY
         LH    R9,0(R10)             LL VALUE
         B     ANCMDCH               CHECK COMMAND COMPLETE
         EJECT
* THE FOLLOWING ARE PROFILE INTRODUCERS, SO WE JUST SKIP
* OVER THE LLCTF
```

```
         SPACE
AN51     EQU   *                     PROFILE INTRODUCER
AN52     EQU   *                     BASE SUBPROFILE
AN53     EQU   *                     APPLICATION SUBPROFILE
         LA    R9,5                  5 ONLY, FOR LLCTF
         B     ANPRFCH               CHECK TO SEE IF PROFILES COMPLETE
         SPACE
* MANY OF THE DIU OPERANDS WE IGNORE, SO WE GROUP THESE
* TOGETHER AS 'NULL ACTION'
         SPACE
AN54     EQU   *                     3730 SUBPROFILE
AN55     EQU   *                     DISOSS SUBPROFILE
AN56     EQU   *                     5520 SUBPROFILE
AN60     EQU   *                     DOCUMENT NAME
AN62     EQU   *                     OWNER
AN63     EQU   *                     AUTHOR
AN64     EQU   *                     DOCUMENT GCID
AN65     EQU   *                     DOCUMNT TYPE
AN66     EQU   *                     CREATE DATE/TIME
AN67     EQU   *                     LAST CHANGED DATE
AN68     EQU   *                     COPY LIST
AN69     EQU   *                     FILE CABINET REF
AN70     EQU   *                     SUBJECT
AN71     EQU   *                     SYSTEM CODE
AN72     EQU   *                     DOCUMENT SIZE
AN74     EQU   *                     DOCUMENT CLASS
AN75     EQU   *                     DOCUMENT DATE
AN76     EQU   *                     LEVEL 3 PARAMETER SET
AN80     EQU   *                     FILE DATE/TIME
AN81     EQU   *                     OWNERSHIP
AN82     EQU   *                     KEYWORDS
AN83     EQU   *                     EXPIRY DATE
AN84     EQU   *                     OWNER DELEGATE
AN91     EQU   *                     DOCUMENT CONTENT PROFILES ONLY
AN90     EQU   *                     DOCUMENT CONTENT WITH TEXT
         LH    R9,0(R10)            LL VALUE
         B     ANPRFCH               CHECK TO SEE IF PROFILES COMPLETE
         EJECT
************************************************************************
*                                                                      *
*   HERE WE PROCESS THOSE OPERANDS THAT MEAN SOMETHING TO US           *
*                                                                      *
************************************************************************
         SPACE
* SOURCE NAME
AN22     EQU   *
         LH    R9,0(R10)             COLLECT LL VALUE
         LR    R5,R9                 COPY LL VALUE
         S     R5,=F'6'              5 FOR LLCTF + 1 FOR EXECUTE
         EX    R5,MVDLVUSR           PUT IN DPR AREA
         B     ANCMDCH
MVDLVUSR MVC   DPRSID(1),5(R10)      COPY SOURCE NAME
         SPACE
* DOC DISTRIBUTION NAME
AN23     EQU   *
         LH    R9,0(R10)             COLLECT LL VALUE
         MVC   DPRDIS,5(R10)        PUT IN OUT MAP
         B     ANCMDCH
         SPACE
* SOURCE ADDRESS (LOCATION)
AN26     EQU   *
         LH    R9,0(R10)            COLLECT LL VALUE
         LR    R5,R9
         S     R5,=F'6'             5 FOR LLCTF + 1 FOR EXECUTE
         EX    R5,MVDLVLOC          PUT IN OUT MAP
         B     ANCMDCH
MVDLVLOC MVC   DPRDDN(1),5(R10)
         SPACE
* MESSAGE (IF DOC WAS DISTRIBUTED FROM LIBRARY BY ONLINE USER)
AN27     EQU   *
         LH    R9,0(R10)            COLLECT LL VALUE
         LR    R5,R9
         S     R5,=F'6'             5 FOR LLCTF + 1 FOR EXECUTE
         EX    R5,MVDLVMSG          PUT IN DPR BLOCK
         LA    R5,1(R5)             PUT BACK 1 FOR EXECUTE
         STC   R5,DPRMSGL           SET LENGTH IN MAP
         B     ANCMDCH
MVDLVMSG MVC   DPRMSG(1),5(R10)
         SPACE
* EXCEPTION CODE
AN30     EQU   *
         MVC   DPREXCOD,5(R10)      COPY EXCEPTION CODE
         LH    R9,0(R10)            LL VALUE
         B     ANCMDCH             CHECK FOR COMMAND PARMS COMPLETE
         SPACE
* REPLY DATA
AN31     EQU   *
         LH    R9,0(R10)            LL VALUE
         LR    R5,R9
         S     R5,=F'6'             5 FOR LLCTF + 1 FOR EXECUTE
         EX    R5,AN31MVC           PUT IN DPR BLOCK
         B     ANCMDCH              CHECK FOR COMMAND COMPLETE
AN31MVC  MVC   DPRACKR(1),5(R10)
         EJECT
* PROFILE GCID
AN61     EQU   *
         LH    R9,0(R10)            LL VALUE
         MVC   DPRPGC,5(R10)        COPY PROFILE GCID FOR REFILE
         B     ANPRFCH              CHECK FOR PROFILE COMPLETE
         SPACE
* LADN PROFILE OPERAND
AN73     EQU   *
         LH    R9,0(R10)            LL VALUE
         MVC   DPRLAD,7(R10)        COPY DTM PART OF LADN
         B     ANPRFCH              CHECK FOR PROFILE COMPLETE
         EJECT
* TEXT SEGMENT INTRODUCER
AN95     EQU   *
         LH    R9,0(R10)            LL VALUE
         TM    COMDST,COMDSTP       HAVE WE HAD PROFILES ?
         BO    AN95A                YES, PROCESS TEXT
         STH   R9,COMLDIU2          NO, SET SEGMENT LENGTH
         SLR   R0,R0                INITIALIZE ...
         STH   R0,COMLDIU3           .. LENGTH USED
         MVC   DPRDOT,8(R10)        COPY DOC TYPE
         MVC   DPRSYS,10(R10)       AND SYSTEM CODE
         LA    R9,23                INCR FOR LLCTF, ISS, TYPE & CODE
         B     ANPRFCH              CHECK FOR PROFILE COMPLETE
         SPACE
AN95A    EQU   *
* IF THIS IS LAST TEXT SEGMENT, MARK TEXT BUFFER COMPLETE AND QUIT
         TM    5(R10),X'20'          LAST SEGMENT ?
         BZ    AN95C                 YES, FINISH
         SPACE
* WILL SEGMENT FIT IN BUFFER ?
         LR    R7,R9                 ELEMENT LENGTH
         SH    R7,=H'8'              REDUCE FOR LLCTF AND ISS
         LR    R1,R7                 COPY LENGTH FOR LATER
         LH    R0,=H'4088'           MAX DATA IN BLOCK
         SH    R0,0(R2)              CALCULATE SPACE AVAILABLE
         CR    R0,R7                 COMPARE AGAINST THIS BLOCK
         BL    AN95B                 TOO MUCH, WE MUST SPLIT IT
         SPACE
* YES, THERE IS SPACE IN BUFFER
         LH    R5,0(R2)              GET CURRENT LENGTH
         LA    R4,8(R5,R2)           TARGET ADDRESS
         LA    R6,8(R10)             SOURCE ADDRESS
         LR    R5,R7                 TARGET LENGTH
         MVCL  R4,R6                 MOVE TEXT SEGMENT
         SPACE
* AND UPDATE BUFFER USE VALUES
         AH    R1,0(R2)              NEW LENGTH + OLD LENGTH
         STH   R1,0(R2)              UPDATED
         B     AN95R
         SPACE
AN95B    EQU   *
* THERE WAS NOT ENOUGH SPACE IN BUFFER
* R0 = SPACE AVAILABLE  R1 = LENGTH OF DATA
         SPACE
* MOVE AS MUCH AS WILL FIT
         LR    R7,R0                 USE SPACE AVAILABLE
         LH    R5,0(R2)              GET CURRENT LENGTH
         LA    R4,8(R5,R2)           TARGET ADDRESS
         LA    R6,8(R10)             SOURCE ADDRESS
         LR    R5,R7                 TARGET LENGTH
         MVCL  R4,R6                 MOVE TEXT SEGMENT
         SPACE
* WE SET A NEW ELEMENT LENGTH COUNT TO REFLECT THE PSEUDO LLCTF
* CREATED BELOW
         LR    R9,R0                 LENGTH OF TEXT MOVED
*                                    + 8 FOR THE LLCTFISS
*                                    -8 FOR THE LLCTF WE OVERLAY
        SPACE
* NOW WE CREATE AN ARTIFICIAL LLCTF OVER THE LAST 8 BYTES OF THE
* TEXT WE HAVE JUST MOVED.
         LR    R4,R0                 COPY LENGTH MOVED
         LA    R4,0(R4,R10)          8 BYTES B4 END OF TEXT MOVED
         SLR   R1,R0                 R1 HAS RESIDUAL LENGTH
         LA    R1,8(R1)              + 8 FOR LLCTFISS
         STH   R1,0(R4)              STORE LL VALUE
         MVC   2(6,R4),=X'C90381200000' AND CREATE CTFISS
         SPACE
* AND UPDATE BUFFER USE VALUES
         AH    R0,0(R2)              NEW LENGTH + OLD LENGTH
         STH   R0,0(R2)              UPDATED
         SPACE
AN95C    EQU   *                     OUR BUFFER IS FULL
         OI    COMDST,COMDSTD        DOC SEGMENT EXISTS
         SPACE
AN95R    EQU   *
         B     ANEND
         SPACE
         DROP  R2
         EJECT
***********************************************************************
*                                                                     *
*   (ANTAB) TABLE OF VALID CTF VALUES AND PROCESSING ADDRESSES        *
*                                                                     *
***********************************************************************
ANTAB    DS    0D                ALIGN NICELY
         DC    X'C001',X'0000',A(AN10)     PREFIX
         DC    X'CF01',X'0000',A(AN11)     SUFFIX
* COMMANDS
         DC    X'C119',X'0000',A(AN15)     DELIVER COMMAND
         DC    X'C101',X'0000',A(AN16)     ACKNOWLEDGE COMMAND
         DC    X'C10C',X'0000',A(AN17)     SIGNON REPLY
* COMMON COMMAND OPERANDS
         DC    X'C520',X'0000',A(AN20)     IDENTIFIED DATA
         DC    X'C328',X'0000',A(AN21)     CORRELATION
* COMMAND OPERANDS FOR DELIVER
         DC    X'C323',X'0000',A(AN22)     SOURCE NAME
         DC    X'C340',X'0000',A(AN23)     DOC DISTN NAME
         DC    X'C305',X'0000',A(AN24)     ATTRIBUTE LIST
         DC    X'C306',X'0000',A(AN25)     RECIPIENT NAME
         DC    X'C311',X'0000',A(AN26)     ORIGIN NODE ID
         DC    X'C325',X'0000',A(AN27)     MESSAGE
* COMMAND OPERANDS FOR ACKNOWLEDGE
         DC    X'C322',X'0000',A(AN30)     EXCEPTION CODE
         DC    X'C345',X'0000',A(AN31)     REPLY DATA
* COMMAND OPERANDS FOR SIGNON REPLY
         DC    X'C312',X'0000',A(AN40)     FUNCTION SETS
         DC    X'C30D',X'0000',A(AN41)     SIGNON ID
* DOCUMENT PROFILES
         DC    X'CA03',X'0000',A(AN51)     PROFILE INTRODUCER
         DC    X'CA04',X'0000',A(AN52)     BASE SUBPROFILE
         DC    X'CA05',X'0000',A(AN53)     APPLICATION SUB PROFILE
         DC    X'CA70',X'0000',A(AN54)     3730 SUBPROFILE
         DC    X'CA71',X'0000',A(AN55)     DISOSS SUBPROFILE
         DC    X'CA72',X'0000',A(AN56)     5520 SUBPROFILE
* BASE SUBPROFILE OPERANDS
         DC    X'C700',X'0000',A(AN60)     DOCUMENT NAME
         DC    X'C701',X'0000',A(AN61)     PROFILE GCID
         DC    X'C702',X'0000',A(AN62)     OWNER
         DC    X'C704',X'0000',A(AN63)     AUTHOR
         DC    X'C705',X'0000',A(AN64)     DOCUMENT GCID
         DC    X'C706',X'0000',A(AN65)     DOCUMENT TYPE
         DC    X'C707',X'0000',A(AN66)     CREATE DATE/TIME
         DC    X'C708',X'0000',A(AN67)     LAST CHANGED DATE
         DC    X'C709',X'0000',A(AN68)     COPY LIST
         DC    X'C70A',X'0000',A(AN69)     FILE CABINET REF
         DC    X'C70B',X'0000',A(AN70)     SUBJECT
         DC    X'C70C',X'0000',A(AN71)     SYSTEM CODE
         DC    X'C70D',X'0000',A(AN72)     DOCUEMNT SIZE
         DC    X'C720',X'0000',A(AN73)     LADN
         DC    X'C721',X'0000',A(AN74)     DOCUMENT CLASS
         DC    X'C236',X'0000',A(AN75)     DOCUMENT DATE
         DC    X'C770',X'0000',A(AN76)     LEVEL 3 PARAMETER SET
* APPLICATION SUBPROFILE (DLS)
```

```
         DC    X'C740',X'0000',A(AN80)      FILE DATE/TIME
         DC    X'C741',X'0000',A(AN81)      OWNERSHIP
         DC    X'C742',X'0000',A(AN82)      KEYWORDS
         DC    X'C744',X'0000',A(AN83)      EXPIRY DATE
         DC    X'C745',X'0000',A(AN84)      OWNER DELEGATE
* DOCUMENT CONTENT INTRODUCERS
         DC    X'CB01',X'0000',A(AN90)      DOC CONTENT WITH TEXT
         DC    X'CB02',X'0000',A(AN91)      DOC CONTENT WITH PROF
* DOCUMENT CONTENT SEGMENTS
         DC    X'C903',X'0000',A(AN95)      SEGMENT INTRODUCER
         SPACE
ANTABE   EQU    *
ANTABN   EQU (*-ANTAB)/8                    NUMBER OF TABLE ENTRIES
         EJECT
ENDCSECT DS    OH
DFHEISTG DSECT
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR
         COPY  APIREGS
         PRINT ON
         END
```

B.4.10 APIPTDOC

```
PDOC TITLE '*APIPTDOC* - SENDS A DOCUMENT DIU SEGMENT - DISOSS API SAMP*
               LE PROGRAMS'
         PRINT NOGEN
APIPTDOC CSECT
***********************************************************************
*                                                                     *
*   (APIPTDOC) TAKES A PASSED BUFFER FULL OF DATA (MAX LENGTH IS      *
*              4088 BYTES). BUILDS A DIU TEXT SEGMENT INTRODUCER      *
*              AND SENDS THE DIU SEGMENT TO DISOSS                    *
*              IF THE LENGTH IS ZERO THEN THIS IS A 'LAST SEGMENT'    *
*              AND A NULL LAST SEGMENT IS BUILT AND SENT              *
*                                                                     *
*   INPUT PARMLIST:                                                   *
*                  -> TEXT RECORD                                     *
*                  -> TEXT LENGTH (HALFWORD)                          *
*                  -> BUFFER TO BUILD DIU IN                          *
*                                                                     *
*    OUTPUT: COMRETCD HAS RETURN CODE                                 *
*              0 = GOOD RETURN                                        *
*              8 = BAD RETURN FROM DISOSS. COMREASN HAS DISOSS RC     *
*                                                                     *
***********************************************************************
         SPACE
         L     R8,4(R1)           GET APICOM ADDRESS
         USING APICOM,R8
         L     R2,16(R1)          GET BUFFER ADDRESS
         L     R4,8(R1)          GET TEXT ADDRESS
         L     R5,12(R1)          GET ADDR OF TEXT LENGTH
         LH    R5,0(R5)           GET TEXT LENGTH ITSELF
         LR    R9,R2              REMEMBER BUFFER START
         LR    R10,R5             & TEXT LENGTH (DESTROYED BY MVCL)
         SPACE
         LTR   R5,R5              ANY TEXT ?
         BNZ   PTDOC1             YES, BUILD FULL DIU
         SPACE
* ITS A 'LAST SEGMENT' CALL
         BAL   R14,DUSEGL         BUILD NULL LAST SEGMENT
         B     PTDOC2
         SPACE
* ITS A NORMAL CALL TO BUILD A SEGMENT WITH TEXT
PTDOC1   EQU   *
         BAL   R14,DUSEGN         DOC UNIT SEGMENT (NOT LAST)
* MOVE TEXT FOLLOWING DIU SEGMENT INTRODUCER
* R4 -> TEXT. R5 = LENGTH. SET UP ON ENTRY  (R10 = LENGTH ALSO)
         LR    R6,R2              TARGET ADDRESS
         LR    R7,R5              = SOURCE LENGTH, ZERO PAD
         MVCL  R6,R4              MOVE TO BUFFER
         LA    R2,0(R10,R2)       INCREMENT DIU POINTER
         SPACE
* SET VALUES FOR DISOSS
PTDOC2   EQU   *
         ST    R9,COMDPTR         TELL DISOSS WHERE IS DATA
         SLR   R2,R9               GET DATA LENGTH
         STH   R2,COMDLEN           TELL DISOSS
         STH   R2,0(R9)           SET LL VALUE
         SPACE
* CALL DISOSS
```

```
         APICALL APIDISOS,(=CL8'SEND')
         SPACE
* WE RETURN WITH THE RETURN CODE FROM DISOSS
         L     R4,COMRETCD      DISOSS RC
         ST    R15,COMREASN       DISOSS RC IS OUR REASON CODE
         LTR   R15,R15            IS IT OK ?
         BZ    *+8                YES, USE ZERO VALUE AS OUR RC
         LA    R15,8              NO, WE QUIT WITH RC=8
         ST    R15,COMRETCD       STORE RETURN CODE
         B     ENDCSECT           BRANCH ROUND STATIC DATA ETC
         EJECT
         COPY  APIDIUSB           DIU BUILDER SUBROUTINES
         EJECT
ENDCSECT DS    0H
DFHEISTG DSECT
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR
         COPY  APIREGS
         PRINT ON
         END
```

B.4.11 APIPURGE

```
PURG TITLE '*APIPURGE* - PURGE ALL API QUEUED DATA FOR USER - DISOSS AP*
               I SAMPLE PROGRAMS'
         PRINT NOGEN
APIPURGE CSECT
***********************************************************************
*                                                                     *
*   SUBROUTINE TO DO A DISOSS 'PURGE' FOR ALL DATA ASSOCIATED WITH    *
*   THE USER                                                          *
*                                                                     *
*   INPUT: USER NAME (8 BYTES)                                        *
*                                                                     *
***********************************************************************
         SPACE
         L     R8,4(R1)            LOAD APICOM ADDRESS
         USING APICOM,R8
         L     R9,8(R1)            LOAD ADDRESS OF NAME
         SPACE
         MVC   PRGNAME,0(R9)       MOVE IN USER NAME
         MVI   PRGOPT,C'A'         ALL OF IT
         LA    R0,PRG              GET COMMAND DATA
         ST    R0,COMDPTR           & TELL DISOSS
         LA    R0,PRGLEN           GET DATA LENGTH
         STH   R0,COMDLEN           TELL DISOSS
         SPACE
* NOW CALL DISOSS
         APICALL APIDISOS,(=CL8'PURGE')
         ST    R15,COMREASN       SET REASON CODE AS DISOSS RC
         LTR   R15,R15            WAS IT BAD RC ?
         BZ    *+8                NO, USE ZERO AS OUR RC
         LA    R15,8              OTHERWISE WE HAVE RC 8
         ST    R15,COMRETCD
         EJECT
DFHEISTG DSECT
PRG      DS    0F
PRGNAME  DS    CL8                    NAME TO PURGE
PRGOPT   DS    C                      PURGE OPTION
PRGLEN   EQU   *-PRG
         SPACE
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR
         COPY  APIREGS
         PRINT ON
         END
```

B.4.12 APIRECVE

```
RECV TITLE '*APIRECVE* - RECEIVE A RESPONSE DIU SEGMENT - DISOSS API SA*
               MPLE PROGRAMS'
         PRINT NOGEN
APIRECVE CSECT
***********************************************************************
*                                                                     *
*   (RECEIVE) SUBROUTINE TO RECEIVE A RESPONSE FROM DISOSS            *
*                                                                     *
***********************************************************************
         SPACE
         L     R8,4(R1)
         USING APICOM,R8
         SPACE
* NOW CALL DISOSS
         APICALL APIDISOS,(=CL8'RECEIVE')
         SPACE
* CHECK ON DISOSS RC
         ST    R15,COMREASN       SET REASON CODE AS DISOSS RC
         LTR   R15,R15            WAS IT BAD RC ?
         BZ    *+8                NO, USE ZERO AS OUR RC
         LA    R15,8              OTHERWISE WE HAVE RC 8
         ST    R15,COMRETCD
         EJECT
DFHEISTG DSECT
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR
         COPY  APIREGS
         PRINT ON
         END
```

B.4.13 APIRTRVE

```
RTRV TITLE '*APIRTRVE* - RETRIEVE CICS "START" DATA - DISOSS API SAMPLE*
                PROGRAMS'
         PRINT NOGEN
APIRTRVE CSECT
***********************************************************************
*                                                                     *
*   (RETRIEVE) - SUBROUTINE TO PICK UP CICS 'START' DATA              *
*    ON RETURN COMDPTR -> RETURNED DATA                               *
*              COMDLEN = LENGTH OF DATA                               *
***********************************************************************
         SPACE
         L     R8,4(R1)
         USING APICOM,R8
         SPACE
         EXEC CICS HANDLE CONDITION ENDDATA(RETR0)
         MVC   COMDLEN,=AL2(100)    ARBITRARY MAX VALUE (NEED 36)
         EXEC CICS RETRIEVE SET(R5) LENGTH(COMDLEN)
         ST    R5,COMDPTR            SET DATA POINTER FOR CALLER
         B     RETR1                 OK, SET RC VALS
RETR0    EQU   *
         SLR  R0,R0
         STH  R0,COMDLEN      INDICATE NO START DATA
RETR1    EQU   *
         EXEC CICS HANDLE CONDITION ENDDATA
         SLR   R15,R15
         ST    R15,COMRETCD       SET ZERO RETURN CODE
         EJECT
DFHEISTG DSECT
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR
         COPY  APIREGS
         PRINT ON
         END
```

B.4.14 APISGNON

```
SON  TITLE '*APISGNON* - PERFORM API SIGNON - DISOSS API SAMPLE PROGRAM*
               S'
         PRINT NOGEN
APISGNON CSECT
*********************************************************************
* (SIGNON) - BUILDS A 'SIGNON' DIU                                  *
*********************************************************************
        SPACE
        L      R8,4(R1)            APICOM ADDRESS
        USING  APICOM,R8
        L      R2,8(R1)           DIU BUFFER ADDRESS
        LR     R9,R2
        L      R5,12(R1)          ADDRESS SIGNON PARMS
        SPACE
* DIU PREFIX
        LA     R1,=C'01'          SET ADDRESS AND
        LA     R0,2                LENGTH OF CORRELATION DATA
        BAL    R14,DIUPFX         BUILD DIU PREFIX
* COMMAND SEQUENCE - 'SIGNON'
        LR     R6,R2              SAVE ADDR OF CMD SEQUENCE LL
        BAL    R14,SONCMD         BUILD 'SIGNON' COMMAND
* FUNCTION SETS: WE TAKE ALL SOURCE/RECIPIENT FS
        LA     R1,=X'02000202000402000502000802000902000A'
        LA     R0,18              FUNC SETS 2, 4, 5, 8, 9, 10
        BAL    R14,FUNCSETS       INDICATE WHICH FUNCTION SETS
* SIGNON ID
        LA     R1,4(R5)           SIGNON NAME
        SLR    R0,R0
        IC     R0,2(R5)           LENGTH OF NAME
        BAL    R14,SONNAME        SET NAME
* SIGNON PASSWORD
        LA     R1,12(R5)          SIGNON PASSWORD
        SLR    R0,R0
        IC     R0,3(R5)           LENGTH OF PASSWORD
        LTR    R0,R0              IS THERE A PASSWORD ?
        BZ     SIGNON1            NO, SKIP PASSWORD
        BAL    R14,SONPASS        SET PASSWORD
SIGNON1 EQU    *
* TYPE OF DOCUMENTS TO BE RECEIVED
        LH     R0,0(R5)           DOC TYPE.
        LTR    R0,R0              ANY TYPE SPECIFIED ?
        BZ     SIGNON2            NO, SKIP DOC TYPE
        BAL    R14,RCVDOCS        SET DOC TYPES TO RECEIVE
SIGNON2 EQU    *
* SET LENGTH OF COMMAND SEQUENCE SEGMENT
        LR     R0,R2              SET CURRENT POINTER
        SLR    R0,R6              SUBTRACT ADDR OF CMD SEQ LL
        STH    R0,0(R6)           SET CMD SEQUENCE LL
* SET DIU SUFFIX
        BAL    R14,DIUSFX         BUILD DIU SUFFIX
        EJECT
* SET PARMS FOR DISOSS CALL
        ST     R9,COMDPTR         TELL DISOSS WHERE IT IS
        SLR    R2,R9              OBTAIN LENGTH
        STH    R2,COMDLEN         TELL DISOSS ALSO
         SPACE
* NOW CALL DISOSS
```

```
         APICALL APIDISOS,(=CL8'BIND')
         SPACE
* CHECK ON DISOSS RC
         ST    R15,COMREASN        SET REASON CODE
         LTR   R15,R15             WAS IT BAD RC ?
         BZ    *+8                 NO, USE ZERO AS OUR RC
         LA    R15,8               OTHERWISE WE HAVE RC 8
         ST    R15,COMRETCD
         B     ENDCSECT
         EJECT
         COPY  APIDIUSB            DIU BUILDER SUBROUTINES
         EJECT
ENDCSECT DS    0H
DFHEISTG DSECT
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR
         COPY  APIREGS
         PRINT ON
         END
```

B.4.15 APISNOFF

```
SOFF TITLE '*APISNOFF* - PERFORM DISOSS API SIGNOFF - DISOSS API SAMPLE*
                PROGRAMS'
         PRINT NOGEN
APISNOFF CSECT
***********************************************************************
*   (SIGNOFF)                                                         *
*   SUBROUTINE TO SIGN OFF FROM DISOSS                                *
***********************************************************************
         SPACE
         L     R8,4(R1)            APICOM ADDRESS
         USING APICOM,R8
         SPACE
         LA    R0,SNOFF             GET COMMAND DATA
         ST    R0,COMDPTR            & TELL DISOSS
         LA    R0,SNOFFL            GET DATA LENGTH
         STH   R0,COMDLEN            TELL DISOSS
         SPACE
* NOW CALL DISOSS
         APICALL APIDISOS,(=CL8'UNBIND')
         SPACE
* CHECK ON DISOSS RC
         ST    R15,COMREASN        SET REASON CODE FROM DISOSS RC
         LTR   R15,R15             WAS IT BAD RC ?
         BZ    *+8                 NO, USE ZERO AS OUR RC
         LA    R15,8               OTHERWISE WE HAVE RC 8
         ST    R15,COMRETCD
         B     ENDCSECT            BRANCH ROUND STATIC DATA ETC
         EJECT
***********************************************************************
*                                                                     *
*   THIS IS THE SIGNOFF DIU                                           *
*                                                                     *
***********************************************************************
         SPACE
SNOFF    DC    X'0007',X'C00102',C'04'
         DC    X'0005',X'CD0D01'
         DC    X'0005',X'CF0100'
SNOFFL   EQU   *-SNOFF
         EJECT
ENDCSECT DS    0H
DFHEISTG DSECT
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR
         COPY  APIREGS
         PRINT ON
         END
```

B.4.16 APISUFIX

```
SUFX TITLE '*APISUFIX* - SEND A DIU SUFFIX TO DISOSS - DISOSS API SAMPL*
               E PROGRAMS'
         PRINT NOGEN
APISUFIX CSECT
***********************************************************************
*                                                                     *
*   (SUFFIX) SUBROUTINE TO SEND A STANDARD SUFFIX.                    *
*                                                                     *
***********************************************************************
         L     R8,4(R1)            APICOM ADDRESS
         USING APICOM,R8
         SPACE
         LA    R0,SUF1             GET COMMAND DATA
         ST    R0,COMDPTR           & TELL DISOSS
         LA    R0,SUF1L            GET DATA LENGTH
         STH   R0,COMDLEN           TELL DISOSS
         SPACE
         SPACE
* NOW CALL DISOSS
         APICALL APIDISOS,(=CL8'SEND')
         SPACE
* CHECK ON DISOSS RC
         ST    R15,COMREASN       SET REASON CODE FROM DISOSS RC
         LTR   R15,R15            WAS IT BAD RC ?
         BZ    *+8                NO, USE ZERO AS OUR RC
         LA    R15,8              OTHERWISE WE HAVE RC 8
         ST    R15,COMRETCD
         B     ENDCSECT           BRANCH ROUND STATIC DATA ETC
         EJECT
***********************************************************************
*  THIS IS THE STANDARD SUFFIX DATA                                   *
***********************************************************************
         SPACE
SUF1     DC    AL2(5),X'CF0100'                 DIU SUFFIX
SUF1L    EQU   *-SUF1
         EJECT
ENDCSECT DS    0H
DFHEISTG DSECT
         PRINT OFF
         COPY  APICOM
         COPY  APIDPR
         COPY  APIREGS
         PRINT ON
         END
```

C.0 SAMPLE EXECS FOR THE CMS AND PROFS USER

C.1 SAMPLE DBTMENU EXEC

```
&TRACE OFF

************************************************************************
* Sample EXEC2 program to present a menu of DISOSS functions to the    *
* CMS/PROFS user, and invoke appropriate EXECs to process requests.    *
*                                                                      *
************************************************************************

-START
CLEAR
&BEGTYPE -ENDTYP1
                 D I S O S S   T A S K S


  Which Task do you require?

  Send a Document to DISOSS Users           -  1
  Logon to DISOSS/PS                        -  2
  Read in and send a Note to DISOSS Users   -  3
  Receive a Document from DISOSS            -  4
  Quit                                      -  9

  Enter 1, 2, 3, 4 or 9

-ENDTYP1
&READ VARS &ANS
&IF .&ANS =  .  &GOTO -START
&IF &ANS  =  9  &GOTO -EXIT
&IF &ANS  =  1  &GOTO -SENDOC
&IF &ANS  =  2  &GOTO -PTHRU
&IF &ANS  =  3  &GOTO -SNDNOTE
&IF &ANS  =  4  &GOTO -RECEIVE
&GOTO -START

-SENDOC
EXEC DBTSEND
&GOTO -START

-PTHRU
EXEC DBTLOGON
&GOTO -START

-SNDNOTE
-RECEIVE
&IF &ANS EQ 3 &NXT = DBTNOTE
&IF &ANS EQ 4 &NXT = DBTRECV
&TYPE When you see the RDRLIST panel, enter &NXT under the "Cmd",
&TYPE then press PF10.
&TYPE Now press ENTER to continue.
&READ
EXEC RDRLIST
&IF &RETCODE NE 28 &GOTO -START
```

```
&TYPE
&TYPE Press ENTER to continue.
&READ
&GOTO -START

-EXIT
CLEAR
&EXIT
```

C.2 SAMPLE DBTSEND EXEC

```
&TRACE ERR

************************************************************************
* Sample EXEC2 program to take a 1403 print file, encapsulate it in an *
* MVS job and submit to the MVS system for input to DISOSS V3.         *
*                                                                      *
************************************************************************

************************************************************************
* Set default values.
************************************************************************
* &RSCSVM is the name of the RSCS virtual machine.
&RSCSVM   =  RJE
* &VMNODE is the nodename of the VM system.
&VMNODE   =  RALYDPD3
* &MVSNODE is the nodename of the MVS system.
&MVSNODE  =  RALVSMV3
* &OSN is the DISOSS OSN name. DISOSS default is DSVHOST.
&OSN      =  DSVHOST
* &JOBNAME is first 6 characters of desired MVS jobname.
&JOBNAME  =  DBTDOC
* &JOBACCT is accounting information for MVS jobcard.
&JOBACCT  =  P-032007

************************************************************************
* Direct the virtual punch to the MVS system.                          *
************************************************************************
CP SPOOL   PUNCH CONT TO &RSCSVM
CP TAG DEV PUNCH &MVSNODE JOB 10
&DIACMD = &4
&IF .&4 NE .TEST &SKIP 2
    CP SPOOL PUNCH CONT TO *
    &DIACMD = &5

************************************************************************
* Check that the file exists, and decide whether it has to be chopped  *
* up into 80-byte records.                                             *
************************************************************************
&IF .&1 EQ . &GOTO -PROMPT
-CHECK
&FN = &1
&FT = &2
&FM = &3
&IF .&2 EQ . &FT = MEMO
&IF .&3 EQ . &FM = A
&IF .&4 NE . &IF .&DIACMD EQ . &DIACMD = &4
STATE &FN &FT &FM
&IF &RC EQ 0 &GOTO -GOTFILE

-PROMPT
&BEGTYPE 4

Which file is to be sent?     Enter 'filename filetype filemode'.
                          Defaults:  none     MEMO     A
To quit, just press Enter.
&READ ARGS
```

```
&IF .&1 EQ . &GOTO -END
&GOTO -CHECK

-GOTFILE
LISTFILE   &FN &FT &FM (STACK  FORMAT
&IF &RC NE 0 &EXIT &RC
&READ VARS &FN &FT &FM &J &LRECL
LISTFILE   &FN &FT &FM
DROPBUF 0

&IF &LRECL LE 160 &SKIP 2
    &TYPE Print lines must not be more than 160 chars.
    &EXIT 98
&CDREC = 1
&IF &LRECL GT 80 &CDREC = 2

************************************************************************
* Set up the parameter card for the MVS batch program.                 *
************************************************************************
&CASE M

* DSVHOST is the default OSN name in DISOSS.
&OSN = &LEFT OF &OSN 8

* DISOSS userid. Default is the CMS userid.
IDENTIFY (STACK
&READ VARS &CMSUID
&TYPE Enter DISOSS user name for distribution. Default is &CMSUID
&READ STRING &USER
&IF .&USER EQ . &USER = &CMSUID
&USER = &LEFT OF &USER 8

* ' C' indicates 1403 datastream as input.
&INTYPE = &RIGHT OF C 2

* ' 2' indicates L2DCA datastream as output.
&OUTYPE = &RIGHT OF 2 2

* Number of 80-byte card images representing one printline.
&RECSIZE = &CDREC

* 'A' indicates this is record type A (only type defined at present).
&RECTYPE = A

* 'A' indicates this is the only header card in the file.
&PROFLAG = A

* Number of lines per page (optional).
&PAGEL = &BLANK
&PAGEL = &LEFT OF &PAGEL 3

* Number of characters per line (optional).
&PAGEW = &BLANK
&PAGEW = &LEFT OF &PAGEW 3

* Not used.
&DISNAM = &BLANK
&DISNAM = &LEFT OF &DISNAM 8
```

```
* Eyecatcher
&EYECAT = HEADER

* Document name; default is CMS filename.
&TYPE Enter document name (maximum 15 chars.) Default is &FN
&READ STRING &DOCNAM
&IF .&DOCNAM EQ . &DOCNAM = &FN
&DOCNAM = &LEFT OF &DOCNAM 15

* DIA Command required:  D = Request_Distribution,  F = File.
&UPPER VARS &DIACMD
&IF .&DIACMD NE .D   &IF .&DIACMD NE .F   &DIACMD = D

* Filler
&RESER = &LEFT OF &BLANK 21

* Now build the parameter card.
&PARMCD = &CONCAT OF         &OSN     &USER    &INTYPE  &OUTYPE
&PARMCD = &CONCAT OF &PARMCD &RECSIZE &RECTYPE &PROFLAG &PAGEL   &PAGEW
&PARMCD = &CONCAT OF &PARMCD &DISNAM  &EYECAT  &DOCNAM  &DIACMD  &RESER

*************************************************************************
* Punch the MVS JCL and parameter card.                                 *
*************************************************************************
&J = &SUBSTR OF &TIME 7 2
&JOBCARD = &CONCAT OF // &JOBNAME &J &BLANK JOB &BLANK ( &JOBACCT )
&ROUTECD = &CONCAT OF /*ROUTE &BLANK PRINT &BLANK &VMNODE . &CMSUID

&STACK &JOBCARD
&STACK &ROUTECD
&STACK //VMVSAM  EXEC DBTDOCIN
&STACK //DOCIN    DD  *
&STACK &PARMCD
&STACK
EXECIO * PUNCH

*************************************************************************
* Punch the CMS file; either one or two card images per printline.      *
*************************************************************************
-AGAIN
&TRACE OFF
EXECIO 100 DISKR &FN &FT &FM
&EOF = &RC
SENTRIES
&LOOPCT = &RC
&TRACE ERR

&LOOP -ENDLOOP &LOOPCT
&READ STRING &LINE
&IF &CDREC GT 1 &SKIP 2
    EXECIO 1 PUNCH (STRING &LINE
    &GOTO -ENDLOOP
&LIN1 = &SUBSTR OF &LINE 1  80
&LIN2 = &SUBSTR OF &LINE 81 *
EXECIO 1 PUNCH (STRING &LIN1
EXECIO 1 PUNCH (STRING &LIN2
-ENDLOOP
&IF &EOF EQ 0 &GOTO -AGAIN
```

```
EXECIO 1 PUNCH (STRING /*
EXECIO 1 PUNCH (STRING //

************************************************************************
* Tidy up and exit.                                                    *
************************************************************************
-END
CP SPOOL PUNCH NOCONT CLOSE
CP SPOOL PUNCH OFF
CP TAG DEV PUNCH
&EXIT
```

C.3 SAMPLE DBTLOGON EXEC

```
&TRACE ERR

************************************************************************
* Sample EXEC2 program to invoke VM PASSTHRU in order to logon to      *
* DISOSS/PS.                                                           *
*                                                                      *
* This example selects a specific port on the emulated                 *
* 3271, which allows the user interface to be simplified as follows:   *
*    1 - the VTAM LU represented by that port can be logged on         *
*        automatically to CICS by VTAM via the LOGAPPL parameter, so   *
*        the user does not need to log on to CICS explicitly.          *
*    2 - the CICS 'terminal' represented by the VTAM LU can be         *
*        automatically connected to the DISOSS/PS transaction via the  *
*        TRANSID parameter in the CICS TCT. Thus, the user does not    *
*        have to enter 'DMD1' to select DISOSS/PS.                     *
*                                                                      *
* Consequently, all the user should need to know is:                   *
*    1 - Select '2' from the PROFS menu screen, or enter 'DBTLOGON'    *
*        from CMS.                                                     *
*    2 - When the CICS/VS logo appears, press CLEAR.                   *
*    3 - He should then see the DISOSS/PS logon panel.                 *
*    4 - The following VM PASSTHRU functions are available while       *
*        logged on to DISOSS/PS:                                       *
*        - PA1 will suspend the session and return to CMS. 'DBTLOGON'  *
*          will then allow the DISOSS/PS session to be resumed.        *
*        - PA2 will terminate the CICS session, but should not be used *
*          until the DISOSS/PS session has been ended via PF12.        *
*        - PF10 will copy a screen image into file PASSTHRU DATA on    *
*          the A-disk. Useful to take a quick copy of a DISOSS note.   *
*                                                                      *
************************************************************************

* VM PASSTHRU NODE NAME
&NODE    =  RALYSNA

* PORT NUMBER ON EMULATED 3271
&PORT    =  7

* NAME OF PASSTHRU VIRTUAL MACHINE (DEFAULT IS PVM)
&VMID    =  *

* PFKEY TO INVOKE NOTEPAD
&NOTEPFK =  10

* NO. OF LINES TO BE SAVED BY NOTEPAD
&NOTELIN =  24

* NO. OF COLS. TO BE SAVED BY NOTEPAD
&NOTECOL =  80

* KEY TO BE USED FOR TEMPORARY DISCONNECTION FROM PASSTHRU
&DISC    =  PA1

* KEY TO BE USED TO END PASSTHRU SESSION
&END     =  PA2
```

```
* NOW INVOKE VM PASSTHRU
CP SET MSG OFF
DVMUSI  &NODE &PORT &VMID &NOTEPFK &NOTELIN &NOTECOL &DISC &END
CP SET MSG ON

&EXIT
```

C.4 SAMPLE DBTNOTE EXEC

```
&TRACE ERR

************************************************************************
* Sample EXEC2 program to send a PROFS Note to DISOSS.                 *
*                                                                      *
* First part of the process is, using normal PROFS functions, to send  *
* (or resend) the Note to yourself; this puts the note in your virtual *
* reader. This EXEC then reads it on to the A-disk, edits it to a form *
* suitable to DISOSS, and sends it to DISOSS by calling DBTSEND.       *
*                                                                      *
************************************************************************

&FN = DBTNOTE
&FT = NOTEBOOK
&FM = A

* Read in the file and look for first valid line (Subject:)
ERASE           &FN &FT &FM
EXEC RECEIVE &1 &FN &FT &FM  (NOTEBOOK DBTNOTE
EXECIO * DISKR  &FN &FT &FM  (ZONE 1 9 FIND / Subject:/
&READ VARS &START

* Back up to blank line before "Subject".
&START = &START - 1
DROPBUF 0
LISTFILE   &FN &FT &FM (STACK  ALLOC
&IF &RC NE 0 &EXIT &RC
&READ VARS &FN &FT &FM &J &J &NORECS

* Drop last line of file
&NORECS = &NORECS - 1
DROPBUF 0

* Calculate number of lines to copy
&CPYCT = &NORECS - &START
&CPYCT = &CPYCT  + 1

* Copy to new file, getting rid of nasty hex characters
&STACK FE 40  FF 40  00 40
COPY  &FN &FT &FM  DBTNOTE TEMP A  (REP  FROM &START FOR &CPYCT TRA NOPR

* Send lovely new file to DISOSS/PS
EXEC DBTSEND DBTNOTE TEMP A
&TYPE NOTE has been sent to DISOSS
ERASE DBTNOTE NOTEBOOK A
ERASE DBTNOTE TEMP     A
-END
&EXIT
```

C.5 SAMPLE DBTRECV EXEC

```
&TRACE OFF

************************************************************************
* Sample EXEC2 program to take a 1403 print file sent from DISOSS, and *
* read it on to the A-disk for subsequent filing in PROFS.             *
*                                                                      *
* Note: The VM/SP RECEIVE EXEC used here strips 1403 carriage control  *
* off the file when reading it in. A better solution is needed.        *
*                                                                      *
************************************************************************

STATE DBTRECV TEMP A
&TRACE ERR
&IF &RC EQ 0 &GOTO -RECV
    &TYPE File 'DBTRECV TEMP A' already exists.
    &TYPE Do you want to overwrite it? (y/n)
    &READ VARS &YN
    &IF .&YN NE .Y &EXIT 99

-RECV
ERASE            DBTRECV TEMP A
EXEC RECEIVE &1  DBTRECV TEMP A
&TYPE File DBTRECV TEMP A is now on your A-disk
&TYPE Use the PROFS "Soft Copy" facility to store
&TYPE this document in the PROFS data base.

&EXIT
```

GG24-1604
Connecting non-DIA Systems to DISOSS

READER'S COMMENT FORM

You may use this form to communicate your comments about this publication, its organisation, or subject matter, with the understanding that IBM may use or distribute whatever information you supply in any way it believes appropriate without incurring any obligation to you.

Your comments will be sent to the author's department for whatever review and action, if any, is deemed appropriate. Comments may be written in your own language; use of English is not required.

Note: Copies of IBM publications are not stocked at the location to which this form is addressed. Please direct any requests for copies of publications, or for assistance in using your IBM system, to your IBM representative or to the IBM branch office serving your locality.

Possible topics for comment include: Clarity, Accuracy, Completeness, Organisation, Coding, Retrieval, Legibility.

If you would like a reply, please give your name, company, mailing address and date:

____

____

____

____

What is your occupation? ______________________________

Most recent Newsletter associated with this publication: ________________

Thank you for your cooperation.

Reader's Comment Form

Raleigh International Systems Centre
IBM Corporation (985/B622-3)
PO Box 12195
Research Triangle Park
North Carolina 27709
U.S.A.

For the attention of Gordon Hay.

GG24-1604

OFFICE SYSTEMS INTERCONNECTION:
GUIDE TO CONNECTING NON-DIA SYSTEMS TO DISOSS GG24-1604 PRINTED IN THE U.S.A.

IBM®