March 2000

In this issue

3  Modern CICS application development
11  Submit jobs from CICS to batch
19  Monitoring CICS activity – part 2
29  Finding DFHCSD duplicates and DFH$* groups
48  CICS news

© Xephon plc 2000
CICS Update

Published by
Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38342
From USA: 01144 1635 38342
E-mail: trevore@xephon.com

North American office
Xephon/QNA
1301 West Highway 407, Suite 201-405
Lewisville, TX 75077-2150
USA
Telephone: 940 455 7050

Contributions
Articles published in CICS Update are paid for at the rate of £170 ($250) per 1000 words and £90 ($140) per 100 lines of code for original material. To find out more about contributing an article, without any obligation, please contact us at any of the addresses above and we will send you a copy of our Notes for Contributors.

CICS Update on-line
Code from CICS Update can be downloaded from our Web site at http://www.xephon.com/cicsupdate.html; you will need the user-id shown on your address label.

Editor
Trevor Eddolls

Disclaimer
Readers are cautioned that, although the information in this journal is presented in good faith, neither Xephon nor the organizations or individuals that supplied information in this journal give any warranty or make any representations as to the accuracy of the material it contains. Neither Xephon nor the contributing organizations or individuals accept any liability of any kind howsoever arising out of the use of such material. Readers should satisfy themselves as to the correctness and relevance to their circumstances of all advice, information, code, JCL, and other contents of this journal before making any use of it.

Subscriptions and back-issues
A year’s subscription to CICS Update, comprising twelve monthly issues, costs £175.00 in the UK; $270.00 in the USA and Canada; £181.00 in Europe; £187.00 in Australasia and Japan; and £185.50 elsewhere. In all cases the price includes postage. Individual issues, starting with the January 1994 issue, are available separately to subscribers for £16.00 ($23.50) each including postage.

© Xephon plc 2000. All rights reserved. None of the text in this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, without the prior permission of the copyright owner. Subscribers are free to copy any code reproduced in this publication for use in their own installations, but may not sell such code or incorporate it in any commercial product. No part of this publication may be used for any form of advertising, sales promotion, or publicity without the written permission of the publisher. Copying permits are available from Xephon in the form of pressure-sensitive labels, for application to individual copies. A pack of 240 labels costs $36 (£24), giving a cost per copy of 15 cents (10 pence). To order, contact Xephon at any of the addresses above.

Printed in England.
Modern CICS application development

For the most part, CICS applications are very similar to other applications in your system. Hopefully, they pass through the twelve stages of development—conceive, architect, design, edit code, translate, compile, link-edit or bind, create CICS resource definitions, test, debug, deploy, and maintain. The main differences occur where the environment demands a different architecture—for example batch, where the norm is to repeat some set of functions on a whole file, compared with the on-line transactional environment, where the program normally deals with discrete record(s) from a keyed file or row(s) from a relational database table. A CICS application’s life cycle often lasts more than 20 years, so, not surprisingly, most money is spent in changing and maintaining the business function to keep it in line with regulatory or business focus rather than on the initial implementation.

Today, there is a wide range of extremely powerful and versatile tools, which application developers can use to address each of the development phases. This article highlights some of the options, especially within the edit, compile, and debug areas.

ADVANCES IN DEBUGGING ON MAINFRAMES

In the early days of CICS application development, the ultimate debugging tool was the printed hexadecimal (hex) dump! Once the EXEC CICS interface was added in 1976, it gave the CICS developers the ability to build the CICS Execution Debug Facility, more normally known by its transaction name CEDF. This facility allowed the application programmer to view and change program data as it was passed from the application to CICS and on its return to the application from CICS, and also the condition code allowing the testing of the various error paths. What is particularly powerful about CEDF is that it is non-intrusive, so nothing particular has to be done to the application to use it—therefore it can be used immediately on failing production code. The CEDF functionality was increased in Transaction Server 1.2 with the addition of a new transaction identifier, CEDX,
which performs in the same way without having to be anchored to the transaction being debugged.

Fortunately, matters have moved forward since then, especially for the application developers. Through the ’80s, the dump was superseded by a 3270 terminal as a line-by-line device to look at the source and the values of variables, and then by a full GUI running on a desktop system such as Windows NT or OS/2 connected to your application transaction running within CICS.

A MODERN DEVELOPMENT ENVIRONMENT

If you wish to use your mainframe as the development environment, the latest IBM compilers supply all the technology to perform remote edit, compile, and debug within a CICS address space from a workstation running the VisualAge Remote Debugger. You can debug applications interactively using a full-screen interface or in batch mode using a predefined command file. The following IBM high-level language compilers work with this Debugger:

- IBM COBOL for MVS and VM
- IBM C/C++ for MVS and VM
- IBM OS/390 C/C++
- IBM PL/I for MVS and VM
- VisualAge for Java, Enterprise Edition for OS/390 (remote debug only)
- IBM COBOL for VSE/ESA
- IBM C for VSE/ESA
- IBM PL/I for VSE/ESA.

The facilities provided include single-stepping through each statement of your transaction, inserting break points, or tracing a specific variable. The Debugger can also be dynamically invoked when an error condition occurs, for example when a transaction abend is about to happen. The Debugger displays the source statement at the point of failure, and provides facilities for diagnosing and correcting the
problem. You can change variables and re-execute the code that previously failed without leaving the debug session.

An excellent Web site can be found on this topic at http://www.S390.ibm.com/dt. This site contains various scenarios, detailing where to perform the various tasks, and identifying each task’s benefits and costs. More importantly, it lays out all the levels of prerequisite software and the TCP/IP and SNA APPC connectivity options.

Another approach to CICS application development is for each developer to be completely independent on a Windows NT or OS/2 workstation using IBM VisualAge COBOL Enterprise Edition 2.2 or IBM VisualAge PL/I Enterprise Edition 2.1. Each of these products contains a VisualAge CICS Enterprise Application Development Version 3.1 (which is functionally equivalent to CICS/ESA Version 4.1 and is Year 2000 compliant).

LANGUAGE CAPABILITY

On the language side, there are many major facilities besides the compiler. Firstly, there is a WorkFrame, a configurable project-oriented application development environment, which increases the effectiveness of all the tools.

There is a context sensitive editor, which is able to automatically parse COBOL, PL/I, C/C++, and Java source code – for example, language keywords, comments, and string literals – and display them in different colours and fonts as the program is being created, so saving extra compiles to find just syntax errors.

The Data Assistant simplifies the job of writing syntactically correct SQL by using a point-and-click method on selected database columns, which will then automatically produce the required SQL statements and an Interactive Debug.

Y2K ANALYSIS AND RESTRUCTURING TOOLS

The Millennium Language Extensions (MLE) are part of the set of Redevelopment Tools particularly aimed at understanding your current
source and providing a Year 2000 analysis by chasing the values returned from known time/date constructs through the code or by looking for matches in identifier names, for example a pattern *YYDDD*. Other maintenance work, resulting from having to upgrade to the COBOL ANSI 85 level, can be performed automatically in most cases. Your source code can also be restructured to improve readability and understanding and also to highlight non-reachable code.

ENTERPRISE VISUALAGE COBOL AND PL/I

These extremely powerful desktop packages each contain VisualAge CICS, which runs on Windows NT and OS/2. It is remarkably simple to install, configure, and use. VisualAge CICS’ dynamic resource definition allows application programmers to add new resources while CICS is running, and share these with other programmers on a LAN. Besides being a simple CICS, equivalent to a CICS mainframe address space, it can also support other configurations:

- **FEPI (Front End Processing Interface)**, allowing interconnection with IMS/ESA and CICS/ESA systems as a secondary LU0 and LU2.
- **TCP3270**, giving native connection of TCP boxes to test and run 3270 applications without having to build/use an SNA network.
- **MRO (Multiple Region Operation)**, supporting multiple CICS images on a desktop box.
- **ISC (Inter Systems Coupling)** with other CICS implementations over SNA LU6.2 and TCP/IP with other VisualAge CICS.

A key ease-of-use characteristic is that all the internal data can be held in System/390 formats, so ASCII to EBCDIC and Intel to System/390 data conversions can be a thing of the past. As a result, already existent user data on CICS hosts is available for use immediately with the remote development environment.

The VisualAge package can also be used to remotely edit/compile/debug on your mainframe so, depending upon the needs and goals of your installation, you have an extremely flexible set of ways to build, debug, and maintain your COBOL and PL/I applications.
CICS CLIENT
The CICS client has existed for quite a few years, executing on DOS, Windows 3.1, Windows 95, Windows NT, AIX, Sun Solaris, and OS/2 and connecting over SNALU6.2, TCP/IP, TCP6.2, and NetBIOS, depending on the operating system. The connectivity options are quite complicated, varying by operating system. They are set out clearly in http://www.software.ibm.com/ts/client.

The client provides four basic interfaces to connect an end-user workstation to a mainframe or distributed CICS:

- The External Call Interface (ECI) is a programmable interface that provides the capability to pass a CICS COMMAREA to a CICS server and receive the reply. Various options are provided for different levels of data integrity and the network interactions parallel those of CICS’ Distributed LINK.
- The External Presentation Interface (EPI) is a programmable interface that allows a user application to intercept CICS 3270 data streams. As we will see later, this interface is the basis for putting a GUI on already existing CICS green-screen applications.
- The CICS 3270 emulator will run 3270 CICS server applications without writing any user code.
- The External Security Interface (ESI) is a programmable interface that allows the passing of the user-id and password to a CICS server application.

These client interfaces can be used from COBOL, PL/I, C/C++, and Java applications. Each language has a VisualAge Assistant to make writing CICS client ECI programs easy with their in-built user-friendly editors, compilers, and debuggers.

Recently, IBM has acquired an application development product, which is now shipped as VisualAge Interspace Version 6.0, to work in the CICS client to CICS mainframe rapid application development space.
VISUALAGE INTERSPACE

VisualAge Interspace allows the programmer to build the client side of the application using the tools listed below:

- Microsoft Visual BASIC
- Microsoft ActiveX
- Any Java Integrated Development Environment (IDE), including VisualAge for Java
- Any C or C++ IDE, including VisualAge C++
- Sybase PowerBuilder.

The server systems can be:

- CICS
- MQSeries
- WebSphere
- VisualAge Generator.

It guarantees the ‘handshaking’ between the front-end and back-end applications and so adds significant productivity throughout the application development life-cycle.

In addition to generating client code, VisualAge Interspace generates skeletons for CICS and MQ in C/C++, COBOL, and Java languages.

SERVICE CATALOG

The core building block of VisualAge Interspace is its catalog, which enables server applications to be turned into re-usable application components. Every message that flows between two programs has a message signature with the name of the program or service and the corresponding set of input and output fields; and it is this signature that is stored in the catalog.

The catalog contains meta-definitions of these message signatures, independent of any particular tool, language, or middleware, thereby
making them highly re-usable. Because these messages are defined in the catalog, VisualAge Interspace can guarantee the ‘handshaking’ between these programs in all its environments.

The catalog can be populated from many sources, including COBOL copybooks and BMS maps. VisualAge Interspace also provides a 3270 ‘sniffer’, which catalogs a complete CICS 3270 flow and data simply by running the application through the emulator provided. This allows the mining of a 3270 application, where there is no access to BMS maps. Moreover, because the structure and syntax of the catalog are documented, it is relatively easy to create custom import and export utilities.

**STUB GENERATOR**

A typical middleware application is composed of building blocks called clients and services. A service is much like a function or procedure or even a ‘method’. A client makes a request to invoke the service. Each client and service has business logic within it; however, surrounding this business logic is a middleware-specific ‘wrapper’ that forms the buffer between the middleware and the business logic. These wrappers can be complex and difficult to code.

Through its stub generator, VisualAge Interspace provides a utility program that automatically creates these wrappers from information held in its catalog. This allows front and back-end developers to concentrate exclusively on their own business logic.

The stub generator is driven by templates that can be customized to reflect application-specific processing, such as error management, security, or logging. Because they are based on these templates, the resulting wrappers have this logic embedded within them. This structure guarantees that architecture and processing will be uniformly enforced across application components.

The stub generator supports a broad range of languages. At the client there’s Java, C/C++, Visual BASIC, and PowerBuilder. At the server there’s C/C++, COBOL, and Java. Stubs created with this utility can be imported into IBM tools such as VisualAge for Java, VisualAge
C++, VisualAge COBOL, and VisualAge Generator to enhance developer productivity in building new server-side applications.

API
The VisualAge Interspace API offers a consistent programming interface, semantically and syntactically similar to the API of the particular development tool being used, and covers the full set of functions of the underlying middleware.

TESTER
Testing in a heterogeneous development environment is often a problem. Developers using PowerBuilder, for example, are unlikely to have the equivalent skills to write a COBOL middleware program or vice versa. If both are being developed in parallel, they are unlikely to proceed exactly in step. VisualAge Interspace addresses these problems elegantly, because its tester allows end-to-end testing without end-to-end knowledge. Through the information in the catalog, the tester automatically generates driver programs on both the front and back end, thereby allowing any developer to complete an end-to-end test. The tester works in a live environment – it uses the front-end tool and the actual middleware and connectivity to identify errors as early as possible in the development process.

SERVICE INTERFACE PAINTER
The Service Interface Painter provides a well-integrated, graphical development environment, and is (apart from the API) the primary interface to VisualAge Interspace, offering features for catalog viewing and management, including importing data, stub generation, and testing.

So in conclusion, I believe CICS application development in all its forms has come a long way from punched cards, a line-by-line TSO editor, and a hex dump!

Andy Krasun
IBM (USA)  © IBM 2000
Submit jobs from CICS to batch

With this program you are able to submit jobs from CICS to batch via a central application, which is called CSSUBMI.

The procedure AFPBATCH, in the job embedded in this program, logs in to TSO in batch, and the job specified by other parameters in the COMMAREA is then submitted to run in batch. This means that the submission of the job, which will run in batch, is carried out by the procedure AFPBATCH.

The JCL is stored in a CA-PANVALET library, but you can also define each job in this program directly or use another product similar to CA-PANVALET.

To avoid deadlocks in batch, an ‘EXEC CICS SYNCPOINT’ concludes the logical unit of work.

Originally, the application was designed to submit jobs for a graphic print by IBM’s AFP, but a more universal use is obviously possible.

**AFPBATCH.TXT**

* * This procedure (AFPBATCH) uses standard TSO/ISPF to
* login to TSO in batch and submit a specified job(stream) via
* the TSO/batch environment.
* *
PROC 2 USERID DEST P1() P2() P3() P4() P5() P6() P7() P8() P9()
ISPEXEC FTOPEN
ISPEXEC FTINCL JOB
ISPEXEC FTCLOSE
EXIT

**CSSUBMI.TXT**

*ASM XOPTS(CICS,NOEDF)*
   PUNCH ' MODE AMODE(31),RMODE(ANY) '  
CSSUBMI TITLE 'Program to submit jobs from CICS to batch'
SPACE
DFHEISTG DSECT
You can call the program via "EXEC CICS XCTL" or "EXEC CICS LINK". *
the COMMAREA has the following construction:

* Column   Content
* 1 -  8   Name of the job, which should be submitted from *
          the CA-PANVALET library. *
*         9   Constant ".". *
* 10 - 12  Column 5 to 7 from the name of the submituser. *
*         13  Constant ",". *
* 14 - 19  Name of the printer(JES)/destination on which *
          the output should be printed. *
* 20 - 649 Optional. Type up to nine parameter which should*
          be changed temporarily in the job(stream) from *
          the CA-PANVALET separated with a comma and *
          start with a comma on column 20. *

There are no low-values allowed in the COMMAREA.

The LUW is finished after calling the CSSUBMI-program (syncpoint) *
and there is no direct notification to the *
client application, so it will be useful to keep the calling *
application recoverable.

The name of the submituser has the following construc- *
tion:
* Column   1   = "T", "V" or "P" for the TEST- or PROD-
           CICS or for a CICS between both ("V"). *
* Column   2 - 4   = Constant "SUB".
* Column   5 - 7   = Three-digit shorthand expression for the *
          project to which the calling application *
          is added.

RACF:

* Allow your CICS-regionuser to submit jobs from the CICS via the *
  racfclass SURROGAT *

************************************************************************
EXEC CICS SYNCPOINT
EXEC CICS ADDRESS COMAREA(R2) RESP(RESPONSE)

SPACE
CLC RESPONSE,DFHRESP(NORMAL)
BNE INVCOMMA
CLC EIBCALEN,=H'19'
BL INVCOMMA
CLC EIBCALEN,=H'649'
BH INVCOMMA
LA R0,COMMAR
LH R1,EIBCALEN
LR R3,R1
MVCL R0,R2
CLI KOMMA1,C'.'
BNE INVCOMMA
CLI KOMMA2,C'.'
BNE INVCOMMA
CLI REST,C' ' 
BNH TRUCOMM
CLI REST,C' ' 
BNE INVCOMMA
CLI REST+1,C'P'
BE TRUCOMM
SPACE
INVCOMMA EQU *
MVC OUTMSG,MSG004
BAS R6,SENDMSG
BAS R6,ABEND
B RETURN
SPACE
TRUCOMM EQU *
LA R8,COMMAR
LH R9,=Y(L'COMMAR)
SPACE
TRUCA100 EQU *
CLI Ø(R8),X'00'
BNE TRUCA200
MVI Ø(R8),X'40'
SPACE
TRUCA200 EQU *
LA R8,1(R8)
BCT R9,TRUCA100
SPACE
EXEC CICS ASSIGN SYSID(SYSID) USERID(USERID) RESP(RESPONSE)
SPACE
CLC RESPONSE,DFHRESP(NORMAL)
BE    TRUASSI
MVC  OUTMSG, MSGØØ5
BAS  R6, SENDMSG
BAS  R6, ABEND
B    RETURN
SPACE

TRUASSI  EQU   *
LA    RØ, JOB
LH    R1, JOBLEN
BCTR  R1, Ø
LA    R8, JOBSTART
LR    R9, R1
MVCL  RØ, R8
MVI   OUTMSG, C'
MVC  OUTMSG+1(L'OUTMSG-1), OUTMSG
MVC  SUBUSER(1), ENVIRON
MVC  SUBUSER+1(3), ¬C'SUB'
MVC  SUBUSER+4(L'USER), USER
MVC  JOB+2(L'USERCL7), USERCL7          CARD Ø1
MVC  JOB+175(L'SUBUSER), SUBUSER        CARD Ø3
MVC  JOB+281(L'SUBUSER), SUBUSER        CARD Ø4
MVC  JOB+355(L'PANMEM), PANMEM          CARD Ø5
MVC  JOB+128Ø(L'USERCL7), USERCL7       CARD 17
MVC  JOB+1288(L'DRUCKER), DRUCKER        CARD 17
CLI   REST, C'
BNH   SPOOLO
BAS  R6, PARM
SPACE

SPOOLO  EQU   *
SPACE
EXEC CICS SPOOLOPEN OUTPUT
   USERID('INTRDR')  
   NODE('INTRDR')   
   TOKEN(TOKEN)     
   RESP(RESPONSE) 
SPACE
CLC   RESPONSE, DFHRESP(NORMAL)
BE    SPOOLWRØ
MVC  OUTMSG, MSGØØ1
BAS  R6, SENDMSG
BAS  R6, ABEND
B    RETURN
SPACE

SPOOLWRØ  EQU   *
LA    R1Ø, JOB
SPACE

SPOOLWRØ  EQU   *
MVC  OUTAREA, Ø(R1Ø)
```
SPACE
SPOOLWR1 EQU *
SPACE
EXEC CICS SPOOLWRITE
  TOKEN(TOKEN)
  FROM(OUTAREA)
  RESP(RESPONSE)
SPACE
CLC  RESPONSE,DFHRESP(NORMAL)
BE   SPOOLWR2
MVC  OUTMSG,MSGØØ2
BAS  R6,SENDMSG
BAS  R6,ABEND
B    SPOOLCLS
SPACE
SPOOLWR2 EQU *
CLC  OUTAREA(5),=C'/*EOF'
BE   SPOOLCLS
LA   R1Ø,8Ø(R1Ø)
B    SPOOLWRØ
SPACE
SPOOLCLS EQU *
SPACE
EXEC CICS SPOOLCLOSE
  TOKEN(TOKEN)
  RESP(RESPONSE)
SPACE
CLC  RESPONSE,DFHRESP(NORMAL)
BE   RETURN
MVC  OUTMSG,MSGØØ3
BAS  R6,SENDMSG
BAS  R6,ABEND
SPACE
RETURN EQU *
CLI  OUTMSG,C' '
BNE  RET999
MVC  OUTMSG,MSG999
MVC  OUTMSG+45(L'SUBUSER'),SUBUSER
BAS  R6,SENDMSG
SPACE
RET999 EQU *
EXEC  CICS RETURN
EJECT
* ****************************************************************** *
*                                                                    *
*              S U B R O U T I N E S                                  *
*                                                                    *
* ****************************************************************** *
```
SPACE

SENDSMG EQU *
MVC EYECATO,EYECATI
MVC OLDEIB,DFHEIBLK
EXEC CICS WRITE OPERATOR TEXT(OUTMSG) RESP(RESPONSE)
CLC RESPONSE,DFHRESP(NORMAL)
BER R6
BAS R6,ABEND
B RETURN
EJECT

ABEND EQU *
EXEC CICS ABEND ABCODE('SUBM')
BR R6
EJECT

PARM EQU *
LA R5,JOB+136Ø
LA R7,REST+1
ST R7,STOR7
XR R8,R8

PARM1000 EQU *
CLI Ø(R7),C',''
BNE PARM2000
LA R8,1(R8)
L R3,STOR7
BCTR R8,0
EX R8,MOVE
LA R7,1(R7)
CLI Ø(R7),C',''
BNE PARM9000
LA R7,1(R7)
ST R7,STOR7
XR R8,R8
LA R5,8Ø(R5)

PARM2000 EQU *
LA R7,1(R7)
LA R8,1(R8)
B PARM1000

PARM9000 EQU *
BR R6

MOVE MVC Ø(Ø,R5),Ø(R3)
EJECT

* ************************************************** *
* *
* C O N S T A N T S *
*
** Space **

MSG001 DC CL60'CSSUBMI-001 Error during SPOOL-OPEN |

MSG002 DC CL60'CSSUBMI-002 Error during SPOOL-WRITE |

MSG003 DC CL60'CSSUBMI-003 Error during SPOOL-CLOSE |

MSG004 DC CL60'CSSUBMI-004 Invalid COMMAREA detected |

MSG005 DC CL60'CSSUBMI-005 Invalid ASSIGN detected |

MSG999 DC CL60'CSSUBMI-999 Submit orderly finished for User xxxxxx* x |

EYECATI DC CL40'*** EYE-CATCHER *** OLD EIB-CONTENT ***

** Space **

JOBSTART DC CL80'//B999999A JOB (0000,00000000,00,000,000).

DC CL80'//* KOMMENTAR-KARTE *

DC CL80'// XSUBXXX,REGION=8000K,CLASS=H, *

DC CL80'// MSGCLASS=X,RESTART=*,USER=XSUBXXX *

DC CL80'//JOBREAD EXEC PGM=PANRETRV,PARM='S9999999,EXPAND'' *

DC CL80'//PANLIB DD DSN=NLV.PANLIB.JCL,DISP=SHR *

DC CL80'//D0010S DD DSN=&&TEMP(JOB), *

DC CL80'// DISP=(NEW,KEEP,KEEP), *

DC CL80'// SPACE=(TRK,(10,10,5)), *

DC CL80'// DCB=(RECFM=FB,LRECL=80) *

DC CL80'//ISPSTART EXEC ISPSTART *

DC CL80'//ISPFILE DD SYSOUT=(A,INTRDR), *

DC CL80'//SYSTSIN DD *

DC CL80'ISPSTART CMD(%AFPBATCH + *

DC CL80'B999999 D99999 *

DC CL80' *

DC CL80' *

DC CL80' *

DC CL80' *

DC CL80' *

DC CL80' *

DC CL80' *

DC CL80' *

DC CL80' *

DC CL80' *

DC CL80' *

DC CL80' *)
Monitoring CICS activity – part 2

This month we conclude the article describing how to keep an on-line and historical view of what’s happened inside CICS.

WHAT TO DO IF YOU ARE GOING TO RECYCLE CICS
This is where the process is not automatic any more. The operator should remember to start a special job right before CICS is brought down. The job is the same daily job that was started by the scheduler.

WHAT TO DO IF CICS WAS CANCELLED
Of course if CICS is cancelled this is not the right time to lose your system/application logs.
In order to prevent it, a different job must be run. This ‘post mortem’ job will look at the data from the non-active CICS. It should be run right after a CICS crash, but not necessarily before a new CICS is started. In order to save some space, I’m only showing the first two steps of the JCL – all the rest are equivalent to the job that is run from the scheduler.

This job takes advantage of the fact that in SDSF after the CRDATE command is used, the DDnames are sorted in the same order as in the job.

```plaintext
//EXEC PGM=SDSF,PARM='++24,131'
//ISFOUT DD SYSOUT=T
//ISFIN DD *
//ISFOUT DD SYSOUT=T
//ISFIN DD *
PREFIX SS50CICP
H
SORT CRDATE D
FIND SS50CICP
++?
FIND DCPMJNA
++S
PRINT ODSN 'SYSO.DCPMJNA' * MOD
PRINT
PRINT CLOSE
//EXEC PGM=SDSF,PARM='++24,131'
//ISFOUT DD SYSOUT=T
//ISFIN DD *
PREFIX SS50*
DA
FIND SS50CICP
++?
SORT DSID D
FIND DCPMJNF
DOWN 1
FIND DCPMJNF
++S
PRINT ODSN 'SYSO.DCPMJNF' * MOD
PRINT
PRINT CLOSE
```
PROGRAMS USED FOR AUTOMATING THE SWITCHING PROCESS

The following programs are used to automate the switching process – xeasm1, xeasm2, and xecto.

Xeasm1

* THIS PROGRAM ENABLES DYNAMIC DEALLOCATION OR ALLOCATION OF
  * EXTERNAL TD-QUEUES.
  * IT IS ACTIVATED BY THE FOLLOWING INPUT
  *    TDYN FREE DDRNAME(DDNAME) (UNALLOC ASSUMED)
  * (OR) TDYN ALLOC DDRNAME(DDNAME) SYSOUT(M) (STATUS SHR ASSUMED)
  * UNLIKE ADYN, IT IS NOT A CONVERSATIONAL TRANSACTION
  * EG IT CAN BE USED WITH MVS MODIFY COMMAND.
  * THE PROGRAM USES DYNAMIC ALLOCATION DIRECTLY.
  * IN THE CICS-SAMPLE-GUIDE THERE ARE EXAMPLES OF HOW TO
  * USE DYALLOC FROM CICS.
  * WITH SOME MODIFICATION IN THE DYNAMIC ALLOCATION TEXT UNITS
  * THIS PROGRAM MAY BE USE TO ALLOC/DE-ALLOC DSNAMES.
  * THIS PROGRAM MUST BE LINKED WITH AMOD AND RMOD 24.
  * THE TRANSACTION THAT STARTS THIS PROGRAM SHOULD BE DEFINED WITH
  * TASKDATAKEY=CICS.

RNAME    EQU   5
RLEN     EQU   6
RWK      EQU   4
RRC      EQU   7
RBAL     EQU   8
R1       EQU   1
R15      EQU   15
DFHEISTG DSECT
ECB99    DS    A
TCB99    DS    A
*
TPADYN3  DFHEIENT CODEREG=3,DATAREG=1Ø,EIBREG=11
*      MVC   MSG(29),MSG1
       MVI   TEXT,C' '
       MVC   TEXT+1(Ø9),TEXT
       LA    RWK,Ø
       STH   RWK,TEXTLEN
       EXEC  CICS RECEIVE INTO(TEXT) LENGTH(TEXTLEN)
       CALL  JHTRT,(TEXTLEN,FREELEN)  * IS IT A FREE CALL
       LTR   R15,R15                  * IF IT DOES GO EXECUTE
       BZ    EXFREE
       CALL  JHTRT,(TEXTLEN,ALLOCLN)  * IS IT AN ALLOCATION
       LTR   R15,R15                  * IF IT DOES GO EXECUTE
       BZ    EXALLOC
*      MVC   MSG(29),MSG5          * ELSE IT IS AN ERROR
       * ACTION NOT SPECIFIED
B     BRTRN
EXFREE  BAL  RBAL,DDSRCH  * GO CHECK FOR DDNAME
LTR  RRC,RRC  * IF NO DDNAME FOUND
BP  NOTIFY  * IT IS AN ERROR
BCTR  RLEN,Ø
EX  RLEN,DDCHK  * GO TO EXECUTE MVC
MVI  CIC99VRB,X'Ø2'  VERB CODE FOR UNALLOCATION
MVC  CIC99TLA,−A(CIC99UTU)  ADDR OF LIST OF UNALLOC
XR  R15,R15
LA  R1,CIC99PTR  SVC 99 PARAMETER LIST ADDRESS
DYNALLOC
CH  R15,H04  CHECK SVC 99 RC
BL  BRTRN  0. GO SEND SUCCESS MESSAGE
BH  FAILED  8-12. GO SEND GENERAL ERROR
CLC  CIC99ERR,H1Ø56  4. IS IT X'420'?
BNE  FAILED  4. NO-SEND GENERAL ERROR
B  NOTOPEN  4. YES-SEND NOT OPEN
EXALLOC  BAL  RBAL,DDSRCH  GO LOOK FOR THE DDNAME
LTR  RRC,RRC  * IF BAD RC FROM DDSRCH
BP  NOTIFY
BCTR  RLEN,Ø
EX  RLEN,DDCHK
MVI  CIC99VRB,X'Ø1'  VERB CODE FOR ALLOCATION
MVC  CIC99TLA,−A(CIC99ATU) ADDR OF LIST OF ALLOC TEXT UNITS
XR  R15,R15
LA  R1,CIC99PTR  SVC 99 PARAMETER LIST ADDRESS
DYNALLOC
LTR  R15,R15
BNZ  FAILED
B  BRTRN
FAILED  MVC  MSG(29),MSG4  ACTION FAILED
B  BRTRN
NOTOPEN  MVC  MSG(29),MSG6  ACTION FAILED - NOT OPEN
B  BRTRN
NOTIFY  CH  RRC,X'H8'
MVC  MSG(29),MSG2  DDNAME NOT SPECIFIED
B  BRTRN
RC12  MVC  MSG(29),MSG3  DDNAME TOO LONG
BRTRN  DS  ØH
EXEC  CICS SEND FROM(MSG) LENGTH(29)
EXEC  CICS RETURN
DDSRCH  CALL  JHTRT,(TEXTLEN,DDNLEN)  LOOK FOR DDNAME KEWORD
LTR  R15,R15  NO DDNAME IN INPUT
BNZ  NONAME
LA  RWK,7(R1)  RWK POINT TO TEXT
LR  RNAME,RWK
BCTR  RWK,Ø
NEXT1  LA  RWK,1(RWK)  RWK POINT TO NEXT CHAR
CLC  Ø(1,RWK),HYP  IS IT THE END OF DDNAME
BNE  NEXT1  NO CHECK NEXT CHAR
SR    RWK,RNAME                        YES CHECK IF VALID LENGTH
LTR   RWK,RWK                          CHECK IF DDNAME EXIST
BZ    NONAME                           ELSE RWK CONTAIN DDNAME LENGTH
CH    RWK,=H'8'                        IF DDNAME IS TOO LONG
BH    TOOLONG                           DDNAME TOO LONG
LR    RLEN,RWK                         RLEN CONTAIN STRING LENGTH
SR    RRC,RRC SET RRC TO ZERO
BR    RBAL RETURN TO CALLER
NONAME   LA    RRC,8                    DDNAME MISSING
BR    RBAL
TOOLONG  LA    RRC,12                   DDNAME TOO LONG
BR    RBAL
TEXTLEN DS H
TEXT DS CL9Ø
FREELEN DC H'4'
FREE DC CL4'FREE'
ALLOCN DC H'5'
ALLOC DC CL5'ALLOC'
DDNLEN DC H'7'
DDNAME DC CL7'DDNAME('
HYP DC CL1')'
MSG DS CL29
MSG1 DC CL29'ACTION SUCCESSFULL '
MSG2 DC CL29'DDNAME NOT SPECIFIED '
MSG3 DC CL29'DDNAME TOO LONG '
MSG4 DC CL29'ACTION FAILED '
MSG5 DC CL29'ACTION NOT SPECIFIED '
MSG6 DC CL29'ACTION FAILED FILE IS OPEN '
HØ4 DC H'4'
H1Ø56 DC H'1056'
* SVC 99 PARAMETER LIST
  CIC99PTR DC X'8Ø',AL3(CIC99RB) REQUEST BLOCK POINTER
  CIC99RB DS ØF REQUEST BLOCK
  DC AL1(2Ø) LENGTH OF REQUEST BLOCK
  CIC99VRB DS X VERB CODE
  DC 2X'Ø' FLAGS
  CIC99ERR DC XL2'Ø' ERROR REASON CODE
  DC XL2'Ø' INFORMATION REASON CODE
  CIC99TLA DS A ADDR OF LIST OF TEXT UNIT PTRS
  DC F'Ø' RESERVED
  DC 4X'Ø' FLAGS FOR AUTHORIZED FUNCTIONS
*  
  CIC99ATU DS ØF ALLOC TEXT UNIT POINTER LIST
  DC X'ØØ',AL3(CICDDNAM)
  DC X'ØØ',AL3(CICSYSOU)
  DC X'8Ø',AL3(CICSPINS)
*  
  CIC99UTU DS ØF UNALLOC TEXT UNIT POINTER LIST
  DC X'8Ø',AL3(CICDDNAM)

* TEXT UNITS
CICDDNAM DC X'001',HL2'1',HL2'8',CL8' DDNAME
CICDNUM EQU +8
CICSYSOU DC X'0018',HL2'1',X'D4' SYSOUT(M)
CICSPINS DC X'8013',HL2'1',X'80' PRINT IMMEDIATELY
DDCHK MVC CICDNUM(O),O(RNAME)
END

Xeasm2

JHTRT START
BEGIN EQR=YES
L R3,0(R1) * LOAD STRING ADDRESS
L R4,4(R1) * LOAD SUBSTRING ADDRESS
LH R5,0(R3) * LOAD LENGTH OF STRING
LH R6,0(R4) * LOAD LENGTH OF SUBSTRING
LA R8,0(R3,R5) * R8 POINT AT END OF STRING
BCTR R5,0 * MAKE LENGTH FOR EX COMMAND
BCTR R6,0 * MAKE LENGTH FOR EX COMMAND
CR R5,R6 * IS STRING SHORTER THEN SUBSTRING ?
BL TOOLENG * YES ? GO TELL CALLER
CH R5,=H'255' * IS LENGTH OVER ALLOWED LENGTH ?
BH TOOLENG * YES ? GO TELL CALLER
CH R6,=H'255' * IS LENGTH OVER ALLOWED LENGTH ?
BH TOOLENG * YES ? GO TELL CALLER
LA R3,2(R3) * LOAD ADDR FIRST CHAR OF STRING
LA R4,2(R4) * LOAD ADDR FIRST CHAR OF SUBSTRING
XC TAB(256),TAB * CLEAR TRT TABLE
XR R1,R1 * CLEAR REGISTER
IC R1,0(R4) * FIRST CHAR OF SUBSTRING
LA R2,TAB * LOAD TAB ADDRESS
LA R2,0(R1,R2) * OFFSET OF FIRST CHAR OF SUBSTRING
STC R1,0(R2) * INSERT INTO TAB FOR TRT
EXTRT DS ØH *
EX R5,TRT * EX TRT
BZ NOTFOUND * NOT FOUND ? GO TO END
CR R1,R8 * TRT STOP AFTER STRING LIMIT ?
BH NOTFOUND * NOT LOW ? NOT FOUND
EX R6,CLC * EX CLC
BE FOUND * YES ? GO TO END
LR R9,R1 * SAVE TRT ADDRESS
SR R1,R3 * R1 = TRT ADVANCE IN BYTES
SR R5,R1 * HOW MUCH LEFT
CR R5,R6 * LEFT < SUBTRING LENGTH
BL NOTFOUND * YES ? NOT FOUND
LTR R5,R5 *
BZ NOTFOUND *
LA R3,1(R9) * POINT NEXT CHAR
B EXTRT *
NOTFOUND DS ØH *
LA R15,8 *
Xecto

******************************************************************
* THIS ROUTINE FREE & ALLOC DDNAME DCPMJN? IN CICS WHEN ?=T L F A  *
* ACTIVATED BY RULE - CICSTDQ                                    *
* AFTER THIS PROGRAM IS RUN - A NEW GENERATION OF CICS ARCHIVE    *
* LOG IS CREATED - AND THE REPLACED LOG IS ARCHIVED INTO TAPE.    
******************************************************************

TRACE ON
MAXCOMMAND 9999
TIMEOUT 9999
LOGON APPLID CICSOLIB SESSID KS01
IFVAR %VTAMRC EQ '48' GOTO CICS_DOWN
CURSOR POS 1 8
IFSCREEN 'WELCOME TO CICS' GOTO SIGN_ON
GETSCREEN

*  
LABEL SIGN_ON
CLEAR
TYPE 'CESN USERID=MCPCTO,PS=%A1'
ENTER
CURSOR POS 1 11
IFSCREEN 'SIGN-ON IS COMPLETE' GOTO START_PROG
GOTO SIGN_NOTOK

*  
LABEL START_PROG
SETVAR %DJN1 DATA 'DJNA'
SETVAR %DJN2 DATA 'DJNF'
SETVAR %DJN3 DATA 'DJNL'
SETVAR %DJN4 DATA 'DJNT'
SETVAR %NO_DJN DATA '4'
SETVAR %DCPMJNX_STAT DATA 'FREE'
GOTO NEXT_DJN

* LABEL NEXT_DJN
  IFVAR %NO_DJN EQ '0' GOTO END_PROG
  IFVAR %NO_DJN EQ '1' GOTO DJN1
  IFVAR %NO_DJN EQ '2' GOTO DJN2
  IFVAR %NO_DJN EQ '3' GOTO DJN3
  IFVAR %NO_DJN EQ '4' GOTO DJN4

* LABEL DJN1
  SETVAR %N_D DATA 'DJNA'
  SETVAR %JN DATA 'JNA'
  GOTO TDQ

* LABEL DJN2
  SETVAR %N_D DATA 'DJNF'
  SETVAR %JN DATA 'JNF'
  GOTO TDQ

* LABEL DJN3
  SETVAR %N_D DATA 'DJNL'
  SETVAR %JN DATA 'JNL'
  GOTO TDQ

* LABEL DJN4
  SETVAR %N_D DATA 'DJNT'
  SETVAR %JN DATA 'JNT'
  GOTO TDQ

* LABEL TDQ
  SETVAR %NO_DJN DATA '%NO_DJN %%MINUS 1'
  SETVAR %ACT DATA 'CLOSE'
  PFØ3
  CLEAR
  TYPE 'CEMT S TDQ(%N_D) CL
  ENTER
  CURSOR POS 1 15
  SETVAR %DJN SCREEN 1 12 4
  CURSOR POS 23 14
  IFSCREEN 'NORMAL' GOTO DO_ADYN
  GOTO TDQ_NOTOK

* LABEL TDQ_NOTOK
  SHOUT TO TSO-$Ø1Ø TEXT 'CTO - CICS TDQ %ACT FOR %DJN IS NOT NORMAL'
  SHOUT TO TSO-$ØØ4 TEXT 'CTO - CICS TDQ %ACT FOR %DJN IS NOT NORMAL'
  GOTO END_PROG

* LABEL DO_ADYN
  PFØ3
  CLEAR
TYPE 'ADYN
ENTER
CURSOR POS 9 1
IFSCREEN 'ENTER' GOTO DO_FREE
GOTO DO_OPEN
*
LABEL DO_FREE
CURSOR HOME
TYPE 'FREE DDNAME(DCPM%JN) UNALLOC
ENTER
CURSOR POS 9 41
IFSCREEN 'ØØØØ' GOTO DO_ALLOC
SETVAR %DCPMJNX_STAT DATA 'FREE'
GOTO RC_NOTOK
*
LABEL RC_NOTOK
SHOUT TO TSO-SØ1Ø TEXT 'CTO - CICS FREE DCPM%JN NOT OK'
IFVAR %DCPMJNX_STAT EQ 'ALLOC' GOTO DO_OPEN
*
LABEL DO_ALLOC
PFØ3
CLEAR
TYPE 'ALLOC DDNAME(DCPM%JN) SYSOUT(M)
ENTER
CURSOR POS 9 41
IFSCREEN 'ØØØØ' GOTO DO_OPEN
SETVAR %DCPMJNX_STAT DATA 'ALLOC'
GOTO RC_NOTOK
*
LABEL DO_OPEN
CURSOR HOME
SETVAR %NUL DATA 
TYPE '%NUL'
ENTER
PFØ3
CLEAR
* CURSOR HOME
TYPE 'CEMT S TDQ(%N_D) OP
ENTER
CURSOR POS 23 14
IFSCREEN 'NORMAL' GOTO NEXT_DJN
SETVAR %ACT DATA 'OPEN'
GOTO TDQ_NOTOK
*
LABEL SIGN_NOTOK
SHOUT TO TSO-SØ1Ø TEXT 'cics sign-on failed'
GOTO END_PROG
*
LABEL CICS_DOWN
SETOGLB %%DOWN%A2 = 48
*

Contributing to *CICS Update*

Although the articles published in Xephon *Updates* are of a very high standard, the vast majority are not written by professional writers, and we rely heavily on our readers themselves taking the time and trouble to share their experiences with others. Many have discovered that writing an article is not the daunting task that it might appear to be at first glance. They have found that the effort needed to pass on valuable information to others is more than offset by our generous terms and conditions and the recognition they gain from their fellow professionals. Often, just a few hundred words are sufficient to describe a problem and the steps taken to solve it.

If you have ever experienced any difficulties with CICS, or made an interesting discovery, you could receive a cash payment, a free subscription to any of our *Updates*, or a credit against any of Xephon’s wide range of products and services, simply by telling us all about it. For a copy of our *Notes for Contributors*, which explains the terms and conditions under which we publish articles, please write to the editor, Trevor Eddolls, at any of the addresses shown on page 2, or e-mail him on trevore@xephon.com.
Finding DFHCSD duplicates and DFH$* groups

This article includes JCL to find duplicate entries in DFHCSD and also a procedure to remove DFH$* groups in your production DFHCSD. Included is a PL/I program to delete such keys, but you could also unload VSAM, edit the unloaded dataset and delete records, and then upload.

CSDDUPL

This is the JCL to find duplicate entries in DFHCSD:

```sql
//TSHVRA JOB (), 'CSDDUPL', CLASS=A, MSGCLASS=X, NOTIFY=TSHVR
//*FIND DUPLICATES IN DFHCSD
//*Ø1-Ø8: NAME OF GROUP
//*13-14: TYPE OF RESOURCE
//* ØXØØØ1 GENERAL
//* ØXØØØ5 IBM PROTECTED GROUP IBMINITIAL
//* ØXØØØ6 GROUP
//* ØXØØØD LIST
//*I5-22: NAME OF RESOURCE
//*EXAMPLE OF OUTPUT:
//* Produto      4AIXØ       ØØØØØ1
//* DFHMRODR     4CICA       ØØØØØB  (B=ØXC2=ØXF2 = 2)
//* DFH$FILA      FILEA      ØØØØØC  (C=ØXC3=ØXF3 = 3)
//* DFHCOMP1     EDFHRTY     ØØØØØD  (D=ØXC4=ØXF4 = 4)
//SORT1   EXEC SORT
//SORTIN  DD DISP=SHR, DSN=CICS41Ø.PX.DFHCSD
//SORTOUT DD SYSOUT=X
//SYSIN   DD *
* FOR RECFM=VB: OFFSETS SHOULD INCLUDE 4 BYTE RDW-FIELD
OPTION  EQUALS
INREC  FIELDS=(1, 26, 3X, C'ØØØØØ1')
INCLUDE COND=((17, 2, CH, NE, X'ØØØ1'), AND,
               (17, 2, CH, NE, X'0005'), AND,
               (17, 2, CH, NE, X'0006'), AND,
               (17, 2, CH, NE, X'000D'))
SORT FIELDS=(17, 10, CH, A)
SUM    FIELDS=(30, 6, ZD)
//
//
```
This is the PL/I program to delete VSAM records:

```
* PROCESS LANGLVL(OS,SPROG),SYSTEM(MVS);
/*HVPVS01:ACTIONS ON VSAMS                                          */
/* AUTHOR:HVIERENDEELS@WWW.POSTMASTER.CO.UK                         */
/* ———————————————————————————————————————————————————————————————— */
/* ACTIONS ON VSAM DATASETS                                        */
/* ———————————————————————————————————————————————————————————————— */
/* MAINTENANCE:                                                     */
/* Ø9.94 COMMANDO MOVE DELETE+MOVE NAAR BACKUP */

HVPVS01: PROCEDURE (PARM) OPTIONS (MAIN);
  DCL $ERRPROG CHAR(16) STATIC INIT('*ERROR* HVPVS01:'),
      $NOTFOUND BIN FIXED STATIC INIT(51),
      $INV_CARD BIN FIXED STATIC INIT(999),
      $ACT_FAIL BIN FIXED STATIC INIT(998),
      $CMDINFORM CHAR(1) STATIC INIT('I'),
      $COMMENT CHAR(1) INIT('*'),
      $DELETE CHAR(1) INIT('D'),
      $MOVE CHAR(1) INIT('M'),
      $PRINT CHAR(1) INIT('P'),
      $SET CHAR(1) INIT('S'),
      $ERR_IND CHAR(1) INIT('E'),  /* ERROR INDICATIE */
      $MESS_KEYLEN CHAR(16) INIT('INVALID KEYLEN'),
      $TRUE BIT(1) STATIC INIT('1'B),
      $FALSE BIT(1) STATIC INIT('0'B);
/* EXTERNAL ENTRIES */
  DCL HVPSHWC EXTERNAL ENTRY OPTIONS(ASM INTER RETCODE);
/*HVPDYNA  DYNALLOC PROG */
DCL HVPDYNA  ENTRY OPTIONS(ASM INTER RETCODE);
/* EXTERNAL ENTRIES END */
DCL   PARM                  CHARACTER (1000) VARYING ;
%INCLUDE BUILTIN;
DCL     SYSIN              FILE RECORD INPUT,
EOF_SYSIN            BIT (1) INIT ('0'B) ;
DCL     CARDIN CHAR(300) VARYING ;
DCL     PTR_CARD POINTER ;
DCL 1   CARDIN_REDEF UNALIGNED BASED(PTR_CARD),
  2    LEN            BIN FIXED,
  2    CARD,
  3    CODE           CHAR (1),
  3    GENKEY         CHAR (1),  /* ' ' G */
  3    RESERVED1      CHAR (14),
  3    DDNAME         CHAR (08),
  3    KEYLEN         PIC '999',
  3    RESERVED2      CHAR (53);
DCL     CARD_TELLER BIN FIXED INIT(0);
DCL     BACKUP     FILE RECORD OUTPUT;
DCL     MASTER     FILE RECORD UPDATE KEYED ENV (VSAM GENKEY);
```
DCL EOF_MASTER BIT (1) INIT ('Ø'B) ;
DCL SYSPRINT FILE PRINT OUTPUT;
DCL ACTUAL_FILENAME CHAR(8) INIT(' '),
    ACTUAL_BACKUPNAME CHAR(8) INIT('BACKUP'),
    FULLKEY CHAR(255),
    MAXCC BIN FIXED(31) INIT(Ø),
    MYRC BIN FIXED(31) INIT(Ø),
    KEYLEN_BI BIN FIXED(15),
    IXRKP BIN FIXED(15) INIT(Ø),
    IXXL BIN FIXED(15) INIT(Ø),
    ONCODE_SAV BIN FIXED,
    DATA_MAXRECL BIN FIXED(31);
DCL GENREC@ POINTER;
DCL GENREC CHAR(8Ø) BASED(GENREC@);
DCL VSAMREC CHAR(*) VARYING CONTROLLED;
DCL ERRCODE BIN FIXED(31) INIT(Ø);
DCL ERRMESS CHAR(8Ø) INIT(' ');
DCL SW_MOVE BIT(1) ;
ON ERROR BEGIN;
    /* 1ØØ9 INVALID FILE OPERATION EG DEL KEY ON ESDS */
    ON ERROR SYSTEM;
    MYRC = ONCODE;
    PUT SKIP EDIT($ERRPROG,MYRC,'RAISED IN',ONLOC)
        (X(1),A,X(1),F(4,Ø),X(1),A,X(1),A);
    IF MYRC=9 THEN DO;
        IF (ERRCODE ¬=Ø & ERRMESS¬='') THEN
            PUT SKIP EDIT('ERRCODE=',ERRCODE,'ERRMESS=',ERRMESS)
                (X(1),A,F(4),X(1),A,A);
        ELSE PUT SKIP EDIT('SEE PRECEDING PUT SKIP LISTS FOR PROGRAM')
            (X(1),A);
    END;
    /* IF SUBROUTINE */
    GOTO L_RETURN;
END;
/* IF SUBROUTINE */
GOTO L_RETURN;
END;
    ON KEY (MASTER) BEGIN;
        ONCODE_SAV = ONCODE;
    END;
    ON UNDEFINEDFILE(MASTER) BEGIN;
        ONCODE_SAV = ONCODE;
    END;
    ON ENDFILE (SYSIN) BEGIN;
        EOF_SYSIN = $TRUE;
    END;
    ON ENDFILE (MASTER) BEGIN;
        EOF_MASTER = $TRUE;
    END;
PROGRAM_START:
PTR_CARD=ADDR(CARDIN);
OPEN FILE(SYSIN);
CALL S_GET_SYSIN_RECORD;
DO WHILE(¬EOF_SYSIN);

IF CARD.CODE=$SET    THEN SETCODE: DO;
   CALL PERFORM_SET;
END SETCODE;
ELSE NOT_SETCODE: DO;
   CALL S_OPEN;
   IF IXKL>KEYLEN_BI THEN CARD.GENKEY='G';
   SW_MOVE=$FALSE;
   IF CARD.CODE=$DELETE THEN CALL PERFORM_DELETE;
   ELSE IF CARD.CODE=$PRINT THEN CALL S_PRINT;
   ELSE IF CARD.CODE=$MOVE THEN DO;
      IF CARD.GENKEY='G' THEN DO;
         PUT SKIP EDIT('E','MOVE REQUIRES GENKEY IN #',CARD_TELLER)
            (A(1),X(1),A(50),X(1),F(4));
         MAXCC=MAX($INV_CARD,MAXCC);
      END;
   ELSE DO;
      SW_MOVE=$TRUE;
      CALL PERFORM_DELETE;
   END;
END;/* MOVE */
ELSE DO;
   PUT SKIP EDIT('E','INVALID CARDCODE IN CARD #',CARD_TELLER)
      (A(1),X(1),A(21),X(1),F(4));
   MAXCC=MAX($INV_CARD,MAXCC);
END;/*INVALID CARD CODE */
END NOT_SETCODE;
CALL S_GET_SYSIN_RECORD;
END;/* DO WHILE(¬EOF_SYSIN)*/
MYRC=MAXCC;
L_RETURN:
ON ERROR SYSTEM; /* AVOID ERROR LOOP */
CALL PLIRETC(MYRC);
RETURN;

/* SUBROUTINES */
S_OPEN:PROC;
   IF ACTUAL_FILENAME ¬= CARD.DDNAME THEN DO;
      IF ACTUAL_FILENAME='' THEN CLOSE FILE(MASTER);
      ACTUAL_FILENAME=CARD.DDNAME;
   OPEN FILE (MASTER) TITLE(ACTUAL_FILENAME);
   /* IF SYSOUT FILE : ØC4 ABEND */
   IF ONCODE_SAV ¬= Ø THEN DO;
      PUT SKIP EDIT('E','OPEN FAILED',ACTUAL_FILENAME,'IN CARD #',CARD_TELLER)
         (A(1),X(1),A(21),X(1),F(4));
      SIGNAL ERROR;
      END;/*IF ONCODE_SAV ¬= Ø */
   CALL S_DSNINFO;
   IF ALLOCATION(VSAMREC)=Ø THEN DO;
      ALLOC VSAMREC CHAR(DATA_MAXRECL);
   END;
ELSE DO;
  IF LENGTH(VSAMREC) < DATA_MAXRECL THEN DO;
    FREE VSAMREC;
    ALLOC VSAMREC CHAR(DATA_MAXRECL);
  END;
END;/*IF ACTUAL_FILENAME ≠ CARD/DDNAME*/
END S_OPEN;
/**
S_DSNINFO:PROC:

DCL 1 HVPSHWCCCL UNALIGNED,
   2 JCLLEN   BIN FIXED(15) INIT(CSTG(JCLPARM)),
   2 JCLPARM,
   3 IPO CHAR(9) INIT('ØØ1ØØØØØØ'),
   3 @SHWCPDS POINTER INIT(ADDR(SHWCPDS));

DCL 1 SHWCPDS ,
   2 SHWNAME  CHAR(44),
   2 DTCSZ    BIN FIXED(31),
   2 IXCSZ    BIN FIXED(31),
   2 DTMLR    BIN FIXED(31),
   2 IXMLR    BIN FIXED(31),
   2 SHWRKP   BIN FIXED(15),
   2 SHWKL    BIN FIXED(15),
   2 RES      CHAR(44),
   2 SHWOUT,
   3 SHWLEN1 BIN FIXED(15) INIT(CSTG(SHWINFOSET)),
   3 SHWLEN2 BIN FIXED(15),
   3 SHWACBP  BIN FIXED(31),
   3 SHWTYP   CHAR(1),
   3 SHWINF   SET ,
   4 SHWINFSET1 CHAR(512):

DCL Ø1 HVPDYNAP,
   Ø2 DYNAIPO ,
   Ø3 IPOIII   CHAR(3) INIT('ØØØ'),
   Ø3 IPOPPP ,
   Ø4 PPP1 CHAR(1) INIT('Ø'),
   Ø4 PPP2 CHAR(1) INIT('Ø'),
   Ø4 PPP3 CHAR(1) INIT('Ø'),
   Ø3 IPOOO00  CHAR(3) INIT('ØØØ'),
   Ø2 DYNADUMM CHAR(3) INIT(''),
   Ø2 DYNAPL   BIN FIXED(15) INIT(STG(HVPDYNAP)),
   Ø2 DYNAPCMD CHAR(1), /*INPUT:A(LLOC)/U(NALLOC)*/
   Ø2 DYNAPDSN CHAR(44), /*INPUT */
   Ø2 DYNAPMBR CHAR(8), /*INPUT */
   Ø2 DYNAPDDN CHAR(8), /*INPUT . OUTPUT: IF INPUT BLANKO */
   Ø2 DYNAPDSP CHAR(4), /*INPUT:SHR/OLD/MOD/NEW */
   Ø2 DYNAPDSRG CHAR(4), /*OUTPUT */
   Ø2 DYNAPLRECL BIN FIXED(15) INIT(-1), /*OUTPUT:IF MEMBER*/
Ø2 DYNAPBLKSI BIN FIXED(15) INIT(-1), /*OUTPUT:IF MEMBER*/
Ø2 DYNAPERRC BIN FIXED(15) INIT(-1), /*OUTPUT*/
Ø2 DYNAPINFC BIN FIXED(15) INIT(-1); /*OUTPUT*/
DYNAPCMD=$CMDINFORM;
DYNAPDSN='';
DYNAPMBR='';
DYNAPDDN=CARD.DDNAME;
DYNAPDSP='SHR';
DYNAPDSORG='';
CALL HVPDYNA(HVPDYNAP);
MYRC=PLIRETV();
IF MYRC¬=Ø THEN DO;
  PUT SKIP EDIT($ERRPROG,'RC=',MYRC,' ON INFORM OF ',CARD.DDNAME)
    (A,A,F(5),A,A);
  SIGNAL ERROR;
END;
SHWNAME=DYNAPDSN;
CALL HVPSHWC(HVPSHWCCL);
MYRC=PLIRETV();
IF MYRC¬=Ø THEN DO;
  PUT SKIP EDIT($ERRPROG,'RC=',MYRC,' ON SHOWCAT ',SHWNAME)
    (A,A,F(5),A,A);
  SIGNAL ERROR;
END;
IXRKP=SHWRKP;
IXKL=SHWKL;
DATA_MAXRECL=DTMLR;
END S_DSNINFO;
/**/
PERFORM_DELETE:PROC;
FULLKEY=LOW(CSTG(FULLKEY));
SUBSTR(FULLKEY,1,KEYLEN_BI)=SUBSTR(CARD.RESERVED2,1,KEYLEN_BI);
IF CARD.GENKEY='G' THEN MYRC=S_DELGENKEY(KEYLEN_BI);
ELSE DELETE FILE (MASTER) KEY (FULLKEY);
IF ONCODE_SAV = Ø THEN DO;
  PUT SKIP EDIT('I','DELETED      ',ACTUAL_FILENAME,
    'CARD #',CARD_TELLER)
    (A(1),X(1),A(21),A(Ø8),A(21),X(1),F(4));
END;
ELSE IF ONCODE_SAV = $NOTFOUND THEN DO;
  PUT SKIP EDIT('W','NOTFOUND     ',ACTUAL_FILENAME,
    'IN CARD #',CARD_TELLER)
    (A(1),X(1),A(21),A(Ø8),A(21),X(1),F(4));
  ONCODE_SAV=Ø;
END;/*IF ONCODE_SAV ¬= Ø */
ELSE IF ONCODE_SAV ¬= Ø THEN DO;
  PUT SKIP EDIT('E','DELETE FAILED',ACTUAL_FILENAME,
    'IN CARD #',CARD_TELLER)
    (A(1),X(1),A(21),A(Ø8),A(21),X(1),F(4));
  MAXCC=MAX($ACT_FAIL,MAXCC);
ONCODE_SAV=Ø;
RETURN;
END;/*IF ONCODE_SAV ¬= Ø */
END;/* PERFORM_DELETE*/
/**/

S_DELGENKEY: PROC(KEYLEN_BI) RETURNS(BIN FIXED(31));

DCL KEYLEN_BI BIN FIXED(15);
DCL DEELKEY VARYING CHAR(KEYLEN_BI) ;
DCL MYRC BIN FIXED(31);
EOF_MASTER=$FALSE;
DEELKEY=FULLKEY;

IF KEYLEN_BI>Ø THEN

/*READ FILE (MASTER) KEY(DEELKEY) SET(GENREC@)*/
READ FILE (MASTER) KEY(DEELKEY) INTO(VSAMREC);
ELSE

/*READ FILE (MASTER) SET(GENREC@)*/
READ FILE (MASTER) INTO(VSAMREC);
IF ONCODE_SAV = Ø THEN;
ELSE IF ONCODE_SAV = $NOTFOUND THEN DO;

/*PUT SKIP EDIT('W','NOTFOUND G ',ACTUAL_FILENAME,
 'IN CARD #',CARD_TELLER)
(A(1),X(1),A(21),A(Ø8),A(21),X(1),F(4))
*/
RETURN(MYRC);
ENDIF;/*IF ONCODE_SAV ¬= Ø */
ELSE DO;
    ERRCODE=ONCODE_SAV;
    ERRMESS='READ GENKEY'||DEELKEY;
    SIGNAL ERROR;
END;

DO WHILE(¬EOF_MASTER &
    SUBSTR(VSAMREC,IXRKP+1,KEYLEN_BI)=DEELKEY);
/*SUBSTR(GENREC,IXRKP+1,KEYLEN_BI)=DEELKEY)*/
/*FULLKEY=SUBSTR(GENREC,IXRKP+1,IXKL)*/
FULLKEY=SUBSTR(VSAMREC,IXRKP+1,IXKL);
PUT SKIP EDIT('I','WILL BE DELETED:',FULLKEY)
    (A(1),X(1),A(16),A(54) );
IF SW_MOVE THEN DO;
    WRITE FILE(BACKUP) FROM(VSAMREC);
END;
DELETE FILE (MASTER) KEY (FULLKEY);
IF ONCODE_SAV ¬=Ø THEN DO;
    ERRCODE=ONCODE_SAV;
    ERRMESS='DELETE '||DEELKEY;
    SIGNAL ERROR;
END;
/*READ FILE (MASTER) SET(GENREC@)*/
READ FILE (MASTER) INTO(VSAMREC);
IF ONCODE_SAV ¬=Ø THEN DO;
ERRCODE=ONCODE_SAV;
ERRMESS='READ GENKEY'||DEELKEY;
SIGNAL ERROR;
END;
END;/*DO WHILE(SUBSTR(VSAMREC,IXRKP+1,KEYLEN_BI)=DEELKEY)*/
RETURN(MYRC);
END_S_DELGENKEY;
/**/
S_PRINT: PROC RETURNS(BIN FIXED(31));

DCL RECORDS_READ BIN FIXED(31) INIT(0);

ONCODE_SAV=0;
EOF_MASTER=$FALSE;
PUT SKIP EDIT(' BEGIN OF LIST OF KEYS.') (A);
/* POSITIONERING IS NIET ECHT NODIG DENK IK */
/* JE KAN OOK DIRECT SEQ. BEGINNEN LEZEN */
/* READ VAN ØØØØ ZET BESTANDSPONTET TERUG NAAR BEGIN */
IF IXKL>0 THEN DO;
  FULLKEY=LOW(IXKL);
  READ FILE (MASTER) KEY(FULLKEY) SET(GENREC@);
  IF ONCODE_SAV  =$NOTFOUND THEN DO;
    ONCODE_SAV=Ø;
    READ FILE (MASTER) SET(GENREC@);
  END;
END;
ELSE DO;
  READ FILE (MASTER) SET(GENREC@);
END;
DO WHILE(¬EOF_MASTER);
  IF ONCODE_SAV  ¬= Ø THEN DO;
    PUT SKIP DATA(ONCODE_SAV);
    SIGNAL ERROR;
  END;
  RECORDS_READ=RECORDS_READ+1;
  PUT SKIP EDIT(SUBSTR(GENREC,IXRKP+1,IXKL)) (A);
  READ FILE (MASTER) SET(GENREC@);
END;/*DO ¬EOF*/
PUT SKIP EDIT(' END   OF LIST OF KEYS.') (A);
PUT SKIP EDIT(' RECORDS READ:',RECORDS_READ) (A,F(9));
END S_PRINT;
/**/
S_GET_SYSIN_RECORD:PROC;
DCL TEMPBUF CHAR(300) VARYING;
DCL KEY_BYTES_READ BIN FIXED;
DCL BYTES_READ BIN FIXED INIT(0);
DCL CARDIN_POS BIN FIXED INIT(1);

L_READ:
CARDIN='';
READ FILE (SYSIN) INTO (TEMPBUF);
IF ¬EOF_SYSIN THEN DO;
  CARD_TELLER=CARD_TELLER+1;
  IF ADDR(TEMPBUF)->CARD.CODE=$COMMENT THEN GOTO L_READ;
/**/
  IF ADDR(TEMPBUF)->CARD_DDNAME=' ' THEN DO;
    PUT SKIP EDIT('E','NO DDNAME IN CARD #',CARD_TELLER)
    (A(1),X(1),A(50),X(1),F(4));
    SIGNAL ERROR;
  END;/*IF CARD_DDNAME=' ' */
  IF ADDR(TEMPBUF)->KEYLEN='' THEN KEYLEN_BI=Ø;
  ELSE DO;
    IF VERIFY(ADDR(TEMPBUF)->KEYLEN,'0123456789') > Ø THEN DO;
      PUT SKIP EDIT($ERR_IND,$MESS_KEYLEN,'IN CARD',CARD_TELLER)
      (A(1),X(1),A(16),X(1),A(7),X(1),F(4));
      SIGNAL ERROR;
    END;/*IF VERIFY(KEYLEN,'0123456789')*/
    KEYLEN_BI=ADDR(TEMPBUF)->KEYLEN;
    IF KEYLEN_BI>256 THEN DO;
      PUT SKIP EDIT($ERR_IND,$MESS_KEYLEN,'IN CARD',CARD_TELLER)
      (A(1),X(1),A(16),X(1),A(7),X(1),F(4));
      SIGNAL ERROR;
    END;
    KEYLEN<>''
    KEY_BYTES_READ=LENGTH(TEMPBUF)-27;
  DO WHILE(KEY_BYTES_READ<KEYLEN_BI & ¬EOF_SYSIN);
    CARDIN=CARDIN||SUBSTR(TEMPBUF,1,LENGTH(TEMPBUF));
    READ FILE (SYSIN) INTO (TEMPBUF);
    IF ¬EOF_SYSIN THEN KEY_BYTES_READ=KEY_BYTES_READ+LENGTH(TEMPBUF);
  END;/* */
  IF KEY_BYTES_READ<KEYLEN_BI THEN DO;
    PUT SKIP EDIT($ERRPROG,'EOF ON SYSIN WHILE READING FOR ',
    ' KEYBYTES|')
    (A,A,F(5),A);
    SIGNAL ERROR;
  END;
  CARDIN=CARDIN||SUBSTR(TEMPBUF,1,LENGTH(TEMPBUF));
  END;/*IF ¬EOF_SYSIN */
END S_GET_SYSIN_RECORD;
/***/
PERFORM_SET:PROC;
DCL  1   CARDIN_SET  UNALIGNED BASED(PTR_CARD),
  2   LEN           BIN FIXED,
  2   CARD,
  3   CODE           CHAR (1),
  3   WHAT           CHAR (8),
  3   RESERVED1      CHAR (Ø7),
  3   DDNAME         CHAR (Ø8);
  IF CARDIN_SET.WHAT='BACKUP' THEN SET_BACKUP: DO;
    IF ACTUAL_BACKUPNAME ¬= CARD.DDNAME THEN DO;
      CLOSE FILE(BACKUP);
      ACTUAL_BACKUPNAME=CARDIN_SET.DDNAME;
    END;
  END;/*IF ¬EOF_SYSIN */
END S_GET_SYSIN_RECORD;
/***/
PERFORM_SET:PROC;
OPEN FILE(BACKUP) TITLE(ACTUAL_BACKUPNAME);
END SET_BACKUP;
ELSE SET_UNKNOWN: DO;
    PUT SKIP EDIT('E','INVALID SET CODE IN CARD #',CARD_TELLER)
        (A(1),X(1),A(5Ø),X(1),F(4));
    SIGNAL ERROR;
END SET_UNKNOWN;
END PERFORM_SET;
/**/
END HVPVSØ1;

HVJVSØ1
The following is an example of JCL to execute HVPVSØ1:

//TSHVRA  JOB (),'HVJVSØ1',CLASS=A,MSGCLASS=X,NOTIFY=&SYSUID
/** USE CAPS OFF BECAUSE OF KEY VALUES IN SYSIN
//VSPLI   EXEC PGM=HVPVSØ1
/** TSHVR.CMPLNK.TOOLS(HVPVSØ1)
/** TSHVR.SOURCE.TOOLS(HVPVSØ1)
//STEPLIB DD DISP=SHR,DSN=TSHVR.PGM.TOOLS
//SYSPRINT DD SYSOUT=X,OUTLIM=2ØØØØ
/**BACKUP:LRECL SHOULD BE GREAT ENOUGH TO RECEIVE RECORD
/** BACKUP DD DISP=MOD,DSN=TSHVR.OUTPUT#8
//KSDS    DD DISP=SHR,DSN=TSHVR.TEST.DFHCSD#
//SYSIN   DD *
*POS1:*COMMENT D=DELETE P=PRINT M=MOVE TO BACKUP
*  S=SET
*POS2:G=THIS IS A GENKEY GENERIC
*POS17:DDNAME OF VSAM
*POS25-POS27:GIVEN KEYLEN
*POS28:KEY IF KEY DOES NOT FIT IN RECORD: CONTINUE ON NEXT CARD.
*Ø  1  2  22
*2  7  5  78
*  DDDDDDDLLLLKKKKKKKKKKK.. may be continued on next card
SBACKUP BACKUP
MG    KSDS  ØØ8DFH$IVPL
P     KSDS
/*
 */
/**
/**
DG    KSDS  ØØ1c
P     KSDS
D     KSDS 999//TSHVRA  JOB
() ,'HVJVSØ1',CLASS=A,MSGCLASS=X,NOTIFY=
TSHVR
The following is the SHOWCAT macro required by HVPVS01:

* HVPSHWC : GET INFO ON VSAM VIA SHOWCAT
  *
  * DOEL  : IIVDTCSZ IIVIXCSZ IIVDTMLR IIVIXMLR IIVIXRKP IIVIXKL
  * INPUT :
  *  PARMS: IIIPPPOOONAME
  *   III INPUT
  *    ØØØ NAME TAKE VALID NAME
  *    ØØ1 NAME IS 4 LONG AND TAKE POINTER FILLED BY SHWCPDS
  *     PPP PROCESSING
  *    ØØ SHOWCAT ON DATA INDEX
  *    ØØ1 SHOWCAT ENKEL SHOWCAT ON NAME
  *     OOO OUTPUT
  *    ØØ OUTPUT FROM SHOWCAT IN REXX
  *    ØØ1 RETURN WITH SHWCPDS
  * FILES:
  * OUTPUT:
  * FILES :
  * LAYOUT:
  * WIJZIGINGEN:
  * ABENDS :
  * | OPM | :
  * ARGUMENT (OA PARMLIST) FOR SHOWCAT MUST FOLLOW BELOW
  * OP MOMENT FOR SHOWCAT R13 MUST POINT NEAR SA BELOW
  * REG EQU
  SVCREGØ EQU RØØ   DO NOT TAKE NEAR BASE OF INDEX
  SVCREG1 EQU RØ1
  SVCREG2 EQU RØ2
  BASER   EQU RØ3   BASE REGISTER
  SP      EQU RØ4   STACK POINTER
  WORKØ1 EQU RØ5
  WORKØ2 EQU RØ6
  WORKØ3 EQU RØ7
  WORKØ4 EQU RØ8
  R_SHWCPDS EQU R1Ø
  R_PRMDS EQU R11
  PLITCAR EQU R12   PL/1 TCA
  R_SA    EQU R13   SAVE AREA OF CALLER / OWN SAVE AREA
R_RA    EQU R14 RETURN ADDRESS
R_EP    EQU R15 ENTRYPOINT / RC

* OTHER EQUS
OBJCLUS EQU C'C'             CLUSTER OBJECT
OBJDATA EQU C'D'             DATA COMPONENT
OBJAJIX EQU C'G'             AIX OBJECT
OBJJNX EQU C'I'              INDEX COMPONENT
OBJPATH EQU C'R'             PATH
OBJUPGR EQU C'Y'             UPGRADE SET

COPY HVREGS REGISTER EQUATES
COPY HVRMACS EIGEN MACROS

* PARMLIST DSECT
   DS ØA
   PRMIII DS CL3
   PRMPPP DS CL3
   PRM000D DS CL3
   PRM_END1 DS ØC
   PRMNAME DS CL44 MAY BE DD
   ORG PRM_END1
   PRMSHWCPDS@ DS CL4 ADDRESS

* SHWCPDS DSECT
   DS ØA
   SHWNAME DS CL44 I
   DTCSZ DS F 0
   IXCSZ DS F 0
   DTMLR DS F 0
   IXMLR DS F 0
   IXRKP DS H 0
   IXKL DS H 0
   RES DS CL44 R
   SHWOUT DS ØC W
   SHWLEN1 DS H W
   SHWLEN2 DS H W
   SHWACBP DS A W
   SHWTYPE DS CL1 W
   SHWININFOSET EQU * W
   SHWATTR DS XL1 W
   SHWASSØ DS H W
   SHWASSOCØ DS ØC W
   SHWATYPEØ DS CL1 W
   SHWACØ DS AL3 W
   ORG SHWININFOSET W
   DS X RESERVED W
   SHWWRKP DS H W
   SHWKEYLN DS H W
   SHWCISZ DS FL4 W
   SHWREC DS FL4 W
SHWASS1 DS H W
SHWASSOC1 DS ØC W
SHWATYP1 DS CL1 W
SHWAC1 DS AL3 W
*
HVPShwc1 Csect
ENTRY HVPShwc   PLI CONVENTIe
DC C'HVPShwc'
DC AL1(7)
HVPShwc DS ØH
SAVE (14,12),,HVPShwc1.&SYSTIME..&SYSdate
BALR RØ3.Ø
USING *,RØ3
*
LR RØ3,R15
*
USING HVPShwc1,RØ3
* GETMAIN WORKING STORAGE
LR R_PRMDS,RØ1 SAVE PARM ADDRESS
L RØ6,WS_FIX_LEN
STORAGE OBTAIN,LENGTH=(RØ6),ADDR=(RØ1),
   LOC=(BELOW,ANY)
ST R13,4(0,1) SAVE 13 IN OWN SAVE
ST RØ1,8(0,13) SAVE ADDR. OF SAVE IN PREV. SAVE
LR R13,RØ1
USING WS,R13
ST R_PRMDS,PARMADR
* END GETMAIN WORKING STORAGE
LA SP,STACK
SR SVCREG0,SVCREGØ
ST SVCREGØ,RC
MVI FMSHwpDsd,C'Ø' NO FREEMAIN
* OPEN (SNAPDMP,OUTPUT)
****
* ANY PARMS ?
* JCLLIKE PARMS
L R_PRMDS,Ø(Ø,R_PRMDS)
LTR R_PRMDS,R_PRMDS
BNM L_REXX_PARMS HIGH ORDER BIT OFF
XR WORK04,WORK04
ICM WORK04,B'0011',Ø(R_PRMDS)
LA R_PRMDS,2(Ø,R_PRMDS) BUMP LEN HW. HW IS NOT COUNTED
B L_ANYPARMSE
L_REXX_PARMS EQU *
* REXXLIKE PARMS. PARMADR ALWAYS FILLED BY REXX.
* R01-> DTPTR -> DATA
* LNPTR -> LEN
L R01,PARMADR RELOAD R01
L R_PRMDS,Ø(Ø,R01) POINTER TO DATA POINTER
L WORK04,4(Ø,R01) POINTER TO LENGTH POINTER
LTR R_PRMDS,R_PRMDS
BM L_INV_XP1 HIGH ORDER BIT SHOULD BE OFF
LTR WORK04,WORK04 HIGH ORDER BIT SHOULD BE ON
BNM    L_INV_XP0    IF NOT
L    WORK04,0(0,WORK04)    LENGTH
L    R_PRMDS,0(0,R_PRMDS)    POINTER TO DATA
L_ANYPARMSE    EQU    *
ST    WORK04,#PRMS
ST    R_PRMDS,#PRMS
*    ANY PARMS ?    END
****
*    OPEN (SYSPRINT,OUTPUT)    BEGIN    EQU    *
LA    WORK01,PRM_END1-PARMLIST
CR    WORK04,WORK01
BL    L_BAD_PARML
USING    PARMLIST,R_PRMDS
L    R_PRMDS,#PRMS
CLC    PRMIII,=CL3’ØØØ’
BE    L_III_ØØØ
CLC    PRMIII,=CL3’ØØ1’
BE    L_III_ØØ1
B    L_INVIPO
L_III_ØØ1    EQU    *
***    PRMSHWCPDS@ NAGAAN
LR    R00,WORK04    LENGTH PARM
SR    R00,WORK01    LEN PARM - 9 =    LENGTH ADDRESS
LTR    R00,R00
BNP    L_BAD_PARML
LA    WORK01,L’PRMSHWCPDS@
CR    R00,WORK01
BH    L_BAD_PARML
ICM    R_SHWCPDS,B’1111’,PRMSHWCPDS@
USING    SHWCPDS,R_SHWCPDS
B    L_PROCESS
L_III_ØØ0    EQU    *
***    SHWNAME NAGAAN
LR    R00,WORK04    LENGTH PARM
SR    R00,WORK01    LEN PARM - 9 =    LENGTH NAME
LTR    R00,R00
BNP    L_BAD_PARML
LA    WORK01,L’SHWNAME
CR    R00,WORK01
BH    L_BAD_PARML
STH    R00,#PRMNAME
***    ACQUIRE STORAGE FOR SHWCPDS
LH    WORK01,L_AREA_REST
*    STH    WORK01,L_SHWOUT
STH    WORK01,SHWLEN1    STORAGE NOT YET AVAILABLE
LA    WORK01,L’SHWNAME(0,WORK01)
***    LA    WORK01,SHOWCAT_LSTF_END-SHOWCAT_LSTF(0,WORK01)
STH    WORK01,L_SHWCPDS
STORAGE OBTAIN,LENGTH=(WORK01),ADDR=(R_SHWCPDS),LOC=(BELOW,ANY)
* ST R_SHWCPDS,@SHWCPDS
MVI FMSHWCPS,C'1' FREEMAIN
USING SHWCPDS,R_SHWCPDS
SHOWCAT MF=(B,SHOWCAT_LSTF) INIT AREA
INITB SHWNAME,C'
INITB TYPECI,X'00'
INITB TYPEOK,C'0'
*** SHWNAME INVULLEN
LA WORKØ2,SHWNAME
LA WORKØ1,PRMNAME
LH WORKØ3,#PRMNAME
BCTR WORKØ3,RØØ -1 FOR EXECUTE
EX WORKØ3,L_EXECØ1
*
* LH WORKØ1,L_SHWOUT
* STH WORKØ1,SHWLEN1
*
L_PROCESS EQU *
LA WORKØ2,SHWOUT
LA WORKØ3,SHWNAME
LA RØ1,SHOWCAT_LSTF
SHOWCAT ACB=Ø,AREA=(WORKØ2),NAME=(WORKØ3),MF=(E,(RØ1))
ST R15,RC
LTR R15,R15
BNZ RETURN
CLC PRMPPP,=CL3'001'
BE L_OUTPUT
L_LOOP EQU *
* DEBUG
* LA WORKØ2,SHWOUT
* LH WORKØ3,SHWLEN2
* LA WORKØ3,Ø(WORKØ2,WORKØ3)
* BCTR WORKØ3,Ø
* SNAP DCB=SNAPDMP,ID=1,PDATA=(REGS),STORAGE=((WORKØ2),(WORKØ3))
* DEBUG
* WHICH TYPE ?
CLI SHWTYPE,OBJCLUS
BNE L_AIX
MVI CLOK,C'1'
LH WORKØ1,SHWASSØ
LA WORKØ2,SHWASSOCØ
B L_ASSOCS
L_AIX EQU *
CLI SHWTYPE,OBJAIX
BNE L_DATA
MVI AXOK,C'1'
LH WORKØ1,SHWASSØ
LA WORKØ2,SHWASSOCØ
B L_ASSOCS
L_DATA EQU *
CLI SHWTYPE,OBJDATA
BNE L_INDEX
MVI DTOK,C'1'
MVC DTCSZ,SHWCISZ
MVC DTLR,SHWMREC
LH WORKØ1,SHWASS1
LA WORKØ2,SHWASSOC1
B L_ASSOCS

L_INDEX EQU *
CLI SHWT YPE,OBJINX
BNE L_BAD_TYPE
MVI IXOK,C'1'
MVC IXCSZ,SHWCISZ
MVC IXMLR,SHWMREC
MVC IXRKP,SHWRKP
MVC IXKL,SHWKEYLN
LH WORKØ1,SHWASS1
LA WORKØ2,SHWASSOC1
B L_ASSOCS

L_ASSOCS EQU *
CLI Ø(WORKØ2),OBJCLUS
BNE L_A_AIX
MVC CLCI,1(WORKØ2)
B L_VERHOOG_ASSOC

L_A_AIX EQU * THERE MAY BE MORE AIXS
CLI Ø(WORKØ2),OBJAIX
BNE L_A_DATA
MVC AXCI,1(WORKØ2)
B L_VERHOOG_ASSOC

L_A_DATA EQU *
CLI Ø(WORKØ2),OBJDATA
BNE L_A_INDEX
MVC DTCI,1(WORKØ2)
B L_VERHOOG_ASSOC

L_A_INDEX EQU *
CLI Ø(WORKØ2),OBJINX
BNE L_VERHOOG_ASSOC
MVC IXCI,1(WORKØ2)

L_VERHOOG_ASSOC EQU *
LA WORKØ2,4(Ø,WORKØ2)
BCT WORKØ1,L_ASSOCS

* DEBUG
* LA WORKØ3,TYPeci
* LH WORKØ4,=H'4Ø'
* LA WORKØ4,Ø(WORKØ3,WORKØ4)
* BCTR WORKØ4,Ø
* SNAP DCB=SNAPDMP,ID=2,PDATA=(REGS),STORAGE=((WORKØ3),(WORKØ4))
* DEBUG
* WHAT NEXT? EACH AIX OR CL HAS DATA
XR RØØ,RØØ
CLI DTOK,C'1'
BE L_IX_OK
ICM  RØØ,B'Ø111',DTCI
LTR  RØØ,RØØ
BNZ  L_SHOWCAT1
CLI  IXOK,C'1'  1E SHOWCAT INDEX
BNE  L_INDEKNOOP
BE   L_SHOWCL  1E SHOWCAT ON INDEX

L_XIX_OK  EQU  *   ONLY IF DATA OK
CLI  IXOK,C'1'
BE   L_OUTPUT
ICM  RØØ,B'Ø111',IXCI
LTR  RØØ,RØØ
BNZ  L_SHOWCAT1
CLI  CLOK,C'1'  1E SHOWCAT ON CL AND NO INDEX
BE   L_OUTPUT NO INDEX
*   CLI  AXOK,C'1'  1E SHOWCAT ON AIX AND NO INDEX-> ERROR
*   BE   L_OUTPUT
CLI  DTOK,C'1'  1E SHOWCAT DATA
BNE  L_INDEKNOOP
BE   L_SHOWCL  1E SHOWCAT ON INDEX
L_SHOWCL  EQU  *   SHOWCAT ON CLUSTER
ICM  RØØ,B'Ø111',CLCI
LTR  RØØ,RØØ
BZ   L_SHOWAX
B    L_SHOWCAT1

L_SHOWAX  EQU  *   SHOWCAT ON CLUSTER
ICM  RØØ,B'Ø111',AXCI
LTR  RØØ,RØØ
BZ   L_INDEKNOOP
B    L_SHOWCAT1

L_SHOWCAT1  EQU  *
*   SHOWCAT ON CI
SHOWCAT MF=(B,SHOWCAT_LSTF) INIT AREA
*   LH   WORKØ1,L_SHWOUT
*   STH  WORKØ1,SHWLEN1
LA   WORKØ2,SHWOUT
STCM RØØ,B'Ø111',WORKCI
LA   WORKØ3,WORKCI
LA   RØ1,SHOWCAT_LSTF
L   RØØ,SHWACBP
SHOWCAT ACB=(RØØ),AREA=(WORKØ2),CI=(WORKØ3),MF=(E,(RØ1))
ST   R15,RC
LTR  R15,R15
BNZ  RETURN
BE   L_LOOP

L_OUTPUT  EQU  *
RETURN  EQU  *
*   CLOSE SNAPDMP
*   CLOSE SYSPRINT
***  FREE  STORAGE FOR SHWCPDS
CLI  FMSHWCPDS,C'Ø'  FREEMAIN N/Y
BE   L_156
LH WORK01,L_SHWCPDS
STORAGE RELEASE,LENGTH=(WORK01),ADDR=(R_SHWCPDS)
DROP R_SHWCPDS
L_156 EQU *
L WORK03,RC LOAD RC BEFORE RESTORING R13
L WORK01,SAVEAREA+4 PRECEDING SAVE AREA
L WORK02,WS_FIX_LEN
STORAGE RELEASE,LENGTH=(WORK02),ADDR=(R13)
LR R13,WORK01
LR R15,WORK03
RETURN (14,12),RC=(15)
*
L_EXECØ1 MVC Ø(Ø,WORK02),Ø(WORK01)
*
L_NO_PARMS EQU *
  L R15,=F'4095'
  ST R15,RC
  B RETURN
L_BAD_PARM EQU * PARM LENGTE
  ICM R15,B'1111',PARMLIST
  ICM WORK02,B'1111',PARMLIST+4
  ABEND 4094,DUMP,REASON=(R15)
L_INVIP0 EQU * IN DE KNOOP
  ICM R15,B'1111',PARMLIST
  ICM WORK02,B'1111',PARMLIST+4
  ABEND 4093,DUMP,REASON=(R15)
L_BAD_TYPE EQU * PARM LENGTE
  L R15,=F'4092'
  ST R15,RC
  B RETURN
L_INDEKNOOP EQU * IN DE KNOOP
  L R15,=F'4091'
  ST R15,RC
  B RETURN
L_INV_XPI EQU *
  L R15,=F'4090'
  ST R15,RC
  B RETURN
L_INV_XPO EQU *
  L R15,=F'4089'
  ST R15,RC
  B RETURN
*
* KONSTANTEN
WS_FIX_LEN DC A(WS_FIX_END-WS)
L_AREA_DI DC H'64' LENGTE VVOR D EN I
L_AREA_REST DC H'512' LENGTE VOOR REST VANAF SHWACBP
*
* STATIC STORAGE
**SNAPDMP DCB DDNAME=SNAPDMP,DSORG=PS,MACRF=W,RECFM=VBA,
 ** LRECL=125,BLKSIZE=1632
* DYNAMIC STORAGE
Editor’s note: this article will be concluded in the next issue.

Herman Vierendeels  
Systems Programmer (Belgium)  © Xephon 2000

As a free service to subscribers and to remove the need to rekey the scripts, code from individual articles of CICS Update can be accessed on our Web site, www.xephon.com/cicsupdate.html.

Subscribers need the user-id printed on the envelope containing their Update issue and a copy of the printed issue itself. Once they have registered, any code requested will be e-mailed to them.
Dynasty Technologies has announced Version 4 of its DDE DYNASTY Development Environment.

Among the new features is improved OO functionality with added support for interceptors, enabling developers to intercept messages and add rules and conditions.

There are said to be better facilities for component development, with the addition of the concept of an interface class, enabling developers to break their application into components more easily than before.

What’s more, the bridge between DYNASTY and Rational’s Rose product has been improved to allow greater interaction and changes have been made to speed up the generation process.

I18N support allows the application to have multiple language support and the server processes will act accordingly. This makes it possible, for example, for one application to support Japanese, Swedish, and English clients, with the server transferring data appropriately.

There’s also native support for OS/390, CICS, and DB2 plus Microsoft Repository support during the development process.

Finally, fixed decimal support is expected to enhance the product’s appeal for financial applications development.

For further information contact:
Dynasty Technologies, 101 Redwood Shores Parkway, #200 Redwood Shores, CA 94065, USA.
Tel: (650) 631 5430.

***

Advanced Software Products Group has acquired Command CICS from APT of America (APT Software International). The software translates macro-level code to command-level code.

It translates code immediately at execution time, so there’s no need to allocate resources to rewrite or recompile applications to use command-level calls and conventions needed to migrate to the new CICS Transaction Server.

The software is said to offer the same performance as programs that are manually converted.

For further information contact:
ASPG, 3185 Horseshoe Drive South, Naples, FL 34104, USA.
Tel: (941) 649 1548

***