170 CICS

January 2000

update

In this issue

CICS Update

Published by

Xephon
27-35 London Road
Newbury
Berkshire RG14 1JL
England
Telephone: 01635 38030
From USA: 01144 1635 38030
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.

Printed in England.

Java support in CICS Transaction Server 1.3

INTRODUCTION

CICS Transaction Server for OS/390 Release 1.3 became generally available in March 1999. One of the many enhancements to the base CICS product included in this release of CICS TS was the introduction of support for CICS applications written in the Java programming language. This article describes the background to this new facility within CICS, explains the various stages of application development and implementation for programmers wishing to exploit this new function, and discusses the different options available for object-oriented programming as an interface to CICS.

WHY USE JAVA?

There are many reasons why customers might wish to exploit the Java programming language for CICS applications. One programming model that might be considered is that of front-ending existing CICS business-logic applications with a program written in Java. As will be discussed later, this could exploit the Internet Inter-Orb Protocol (IIOP) to provide a Java-fronted server side to a client/server system communicating via the CORBA specification. This would allow access to existing CICS applications and customer data from a client-based platform, with the benefits of scalability, presentation, and portability that this approach would provide.

Another advantage to developing CICS applications using the Java language is that such applications do not require direct use of the CICS command-level API, as shall be seen later.

SOME TERMINOLOGY

The following terms are referred to in this article, and so warrant definition:

- Class – the code that manipulates the state of something being modelled by a program. A class also defines the data associated with instances of this class, and the interfaces into the code to

manipulate the state of these instances. An example of a class could be BankAccount, which models customer bank accounts and handles the various transactions that can be made against them.

- Method – a defined interface into a class. In the BankAccount class, example methods could be WithdrawMoney, DepositMoney, PrintBalance, etc.
- Object – an instance of a particular class. For example, when a customer opens an account at a bank, a new object of class BankAccount would be created to represent the account. Method calls for the BankAccount class would then be made upon this object, to manipulate its state and return information about it. These would be driven as the customer executed the bank's business logic by withdrawing money, requesting a statement, etc.

  When objects are said to be 'instantiated', it just means that a new instance of an object of a particular class is being created.

COMPILED VERSUS INTERPRETED JAVA

Java was developed by Sun Microsystems as an object-oriented language that first came to prominence for use in coding applets to run on the World Wide Web. Its popularity has increased enormously over the past few years, to the extent that it is becoming pervasive in many aspects of computing, appearing on workstation, mid-range, and mainframe platforms throughout the industry.

'Traditional' Java programs are passed through a compiler that generates platform-neutral bytecodes from the application source. The bytecode files are known as class files. These bytecodes are interpreted by platform-specific Java Virtual Machines (JVMs) that convert the portable Java class files into executable machine code running on a particular computer operating system. For example, source file BankAccount.java would be compiled into bytecodes, producing a class file called BankAccount.class.

One of Java's great strengths is its portability. The class files are interpreted at run-time by the JVMs. It is the responsibility of a JVM

to ensure that this interpretation is handled correctly for the given platform that the JVM is running on. This means that class files are isolated from the specifics of a given operating system, and hence are far more portable than traditional compiled programs.

CICS TS 1.3 provides support for a JVM on MVS running under CICS' control, to interpret Java applications that have been written and compiled into bytecodes. CICS TS 1.3 also supports another Java execution environment. Using the Enterprise Toolkit for OS/390 (ET/390), as supplied with IBM's VisualAge for Java product, Java class files in bytecode format can be further compiled into System/390 machine code and bound into target PDS/E libraries as program objects. These can then be executed under CICS in a similar manner to applications written in other compiled languages such as COBOL and C++.

As with all things in life, there is a trade-off to be made here between the two types of execution environment available for Java applications in CICS TS 1.3. The binding of class files into System/390 machine code by the ET/390 offers a performance improvement over the alternative interpreted approach of running under a JVM, but such a compiled Java program is no longer platform-independent.

Both execution environments for Java applications running in CICS TS 1.3 will be discussed in more detail later in this article.

THE JCICS CLASSES

In addition to providing support for CICS application programming using the Java language, CICS TS 1.3 avoids the need for such programs to include explicit EXEC CICS commands. By providing a class library known as JCICS, support for many of the CICS command-level API core functions can be accessed using supplied methods in the JCICS library. This therefore avoids a requirement to preprocess CICS applications written in Java through the CICS Translator, as you would have to do for COBOL or PL/I command-level applications for example.

The various CICS functions supported either fully or partially by the JCICS classes for Java include Temporary Storage, Transient Data,

Program Control, File Control, Terminal Control (which includes support for a subset of the BMS functions), some Inquire System functions, Assign and Address commands, and Abend processing. CICS functions not supported include Dump, Journal, and Storage services.

When JCICS methods are invoked by CICS application programs written in Java, control passes through some native method code known as the 'Direct To CICS' layer, which converts the JCICS function into a dynamically generated CICS command-level parameter list and associated EXEC CICS request data areas. This is then passed to the CICS Exec Interface program (DFHEIP), which handles the request in the same way as it would an EXEC CICS command being executed by a traditional (ie translated) CICS application program. On completion of the request, control passes back to the Java application via the Direct To CICS layer and the JCICS method once more.

By providing a Java language interface into CICS that can exploit the command-level API without the programmer having to code the equivalent command-level requests, CICS application development is simplified.

DEVELOPING A JAVA APPLICATION

The various stages involved in developing a CICS application written in Java are summarized below.

It is expected that most CICS application development in Java will take place in an Integrated Development Environment (IDE), such as IBM's VisualAge for Java product, running on a workstation. Alternatively, users have the option of using a standard text editor to write their Java applications.

Having developed the Java source for the CICS application, the next stage is to generate the class file (containing the bytecodes) from the source. This compilation step is handled for you by the VisualAge for Java product; alternatively you can use the javac program as supplied with a Java Development Toolkit (JDK). The classpath will need to include the location of the jar file containing the JCICS Java classes, so that method calls to these classes made in the application can be

resolved. CICS TS 1.3 supplies file dfjcics.jar containing these classes. When CICS TS 1.3 is installed, this jar file is placed in the OS/390 Unix System Services Hierarchical File System (HFS).

As a brief aside, Unix System Services was formerly known as OpenEdition. This is the component of System/390 that supplies support for various Unix services under MVS. One such service is a Unix-style directory management system for data storage – the HFS. This is structured in a similar manner to a PC-DOS directory hierarchy. There is one root file system and multiple user file systems are located from that. The root file system can be considered as the base for HFS, and is at the top of the hierarchy of directories. User file systems can themselves contain further sub-directory structures further down the 'tree' of hierarchies. MVS supports this type of structure via a dataset type of HFS, which contains an instance of a hierarchical file system. HFS datasets are single volume datasets; they must be managed by SMS.

Once the application has been compiled into a class file of bytecodes, it could be executed under the MVS JVM environment as supported in CICS TS 1.3. Assuming that the decision has been made to further compile the program into System/390 machine code, however, this class file needs to be transferred to MVS Unix System Services to be processed by the ET/390 binder. This executes the high-performance Java compiler (hpj) to produce a file of machine code from the class file. The ET/390 binder will produce executable program objects and bind them into a target PDS/E library. A PDS/E is a system-managed library that is similar to a PDS but offers various benefits and enhancements over it. For example, PDS/Es provide dynamic compression to free up space within the library, without the need for a manual compression operation. They also allow long-name aliases of the primary names of their members. PDS/Es can only be created when SMS is active and must reside on an SMS-managed volume. PDS/Es are a requirement for CICS TS 1.3 application programs written in Java and processed by the ET/390 binder.

Once the program has been bound into the PDS/E, it can be executed under CICS in the same manner as other compiled application programs. CICS will load the program from the PDS/E as concatenated

in the DFHRPL, and execute it in a Language Environment run-unit. Support for the execution of JCICS method calls from the application is provided by a run-time function included within CICS TS 1.3. ET/390 also provides run-time components that are loaded by CICS to support the Java language environment during the execution of such compiled Java programs.

The CICS TS 1.3 documentation gives fully detailed instructions on the process of developing, building, and executing Java application programs under CICS.

THE JAVA VIRTUAL MACHINE FOR CICS

In addition to 'fully compiled' Java support, which exploits the ET/390 hpj compiler and binder to convert platform-independent Java class files of bytecodes into System/390 machine code, CICS TS 1.3 also supports the traditional Java execution environment; that is, running the bytecodes under a platform-specific JVM and interpreting them at run-time. IBM supplies a JVM on MVS that runs unchanged inside CICS TS 1.3 to perform this interpretation of CICS applications written in Java.

Initially, the same development process would be employed to produce a CICS application program written in Java, as described above in the section *Developing a Java application.*

As before, an IDE or text editor would be used to produce the source file, and this would then be compiled into the interpretable class file of bytecodes. The difference when using the JVM is that there is no need to further process the Java class file that is output from the Java compiler. Having transferred the class file to the MVS Unix System Services HFS, it is available to be loaded by the JVM and interpreted. Note that Java programs that are to be interpreted by the JVM are therefore not loaded from the CICS DFHRPL by CICS directly.

You can include method calls to the various supplied JCICS classes from interpreted JVM programs in the same way that you can from compiled Java applications. In addition, the restrictions on those core Java classes, which are not fully supported in a compiled Java environment, are lifted when using the JVM. This includes functions

such as the Abstract Windowing Toolkit (AWT) class, which provides GUI support.

CICS TS 1.3 has implemented its support for the MVS JVM by means of an enhancement to its dispatching mechanism. This allows CICS application programs written in Java to be executed by a JVM running under its own MVS Task Control Block (TCB). Such TCBs are referred to as open TCBs, and run independently of the CICS quasi-reentrant (QR) TCB that is used for traditional CICS application programs to run under. By providing a unique TCB for a given JVM program, the documented CICS restrictions on the use of certain MVS services that would suspend a TCB are lifted. Such suspension of the QR TCB would have a detrimental effect upon system throughput and performance; by limiting the effect to a particular application running under its own TCB, they do not affect other users of the CICS system.

IIOP SUPPORT

IIOP is an industry standard for distributed programs running across a TCP/IP network. It was developed by the Object Management Group (OMG) as part of the CORBA specification. IIOP requests are processed by an Object Request Broker (ORB) that handles the middleware functions needed to support distributed objects, ie objects with requests being satisfied on distributed platforms.

CICS TS 1.3 supports inbound IIOP requests to CICS application programs written in Java. That is, it allows CICS to act as a server for IIOP requests being sent to it by client programs running on workstations connected to CICS via TCP/IP. CICS TS 1.3 support for IIOP includes an ORB, integrated into CICS, to handle inbound IIOP requests.

In order to be able to communicate between distributed objects using IIOP, information needs to be provided to define the interface between the client and server components. The interface specification used is the Interface Definition Language (IDL). Both the client and the server sides of an IIOP-connected system require an IDL file to define the interface between the two. This file is generated by an IDL compiler (sometimes referred to as an IDL parser). An IDL compiler

is supplied with CICS TS 1.3 to generate the server-side IDL files; vendor packages supplying ORBs for client platforms would supply the equivalent IDL compiler to be run there and generate the client-side IDL files.

SERVICE CONSIDERATIONS

Customers wishing to exploit the Java support in CICS TS 1.3 should ensure that they follow the instructions for installation and requirements for various cross-product dependencies that are documented in the CICS manuals. In particular, the *CICS Program Directory* and *CICS Installation Guide* give guidance on aspects of CICS' Java support installation and run-time requirements.

Of the various cross-product dependencies, the most important are these:

- OS/390 Binder PTFs for APARs OW36582 and OW36049, together with IEBCOPY PDS/E PTFs UW49740 and UW54887, must be installed prior to installing CICS TS 1.3.
- IBM VisualAge for Java PTFs UQ90004 and UQ90006 (for compile-time Java support), and UQ90005 and UQ90007 (for run-time Java support), must be installed to allow CICS TS 1.3's Java support to function correctly.
- Users should ensure that they have the CICS APAR fix for PQ25928 applied to their system. This APAR shipped PTFs UQ29089 and UQ29090.

The Program Directory for CICS TS 1.3 documents these dependencies and requirements in more detail. IBM also maintains an Informational APAR (INFOAPAR number II11860) documenting issues relating to CICS TS 1.3's Java support.

FURTHER OBJECT-ORIENTED SUPPORT IN CICS TS 1.3

In addition to support for application programs written in Java, CICS TS 1.3 also provides another new interface for application programming. Using a similar concept to the JCICS classes, users can

also now write CICS applications in the C++ programming language and invoke CICS service functions via a series of supplied method calls. Known as the C++ Foundation classes, they provide a similar set of method calls to invoke various CICS functions dynamically, without the need to code EXEC CICS commands in the application and have them translated prior to compilation.

Both the JCICS classes and C++ Foundation classes use the native method Direct To CICS layer to interface between their method code and the CICS EXEC Interface program DFHEIP.

ADDITIONAL READING

The CICS TS 1.3 manuals give a detailed explanation of Java application development and implementation under CICS . In addition, IBM has published several Redbooks in this area that are also worth reading. These include *Using VisualAge for Java Enterprise Version 2 to Develop CORBA and EJB Applications* (SRL SG24-5276-00), which discusses Java application development in general, and *Java Application Development for CICS: Base Services and CORBA Client Support* (SRL SG24-5275-00), which is more specific to the CICS environment. Further information on IBM Redbooks can be found on the Web at http://www.redbooks.ibm.com.

Further details of the C++ Foundation classes can be found in the CICS TS 1.3 manual *CICS C++ OO Class Libraries*.

SUMMARY AND CONCLUSIONS

I hope that this article has helped explain the background to CICS application program development using Java in CICS TS 1.3, and the potential that exploiting this new interface into CICS could bring.

Editor's note: readers wishing to discuss the material in this article can contact the author via e-mail at andy_wright@uk.ibm.com. CICS is a registered trademark of IBM.

Andy Wright
CICS Change Team Programmer
IBM (UK)

CICS region data cloning

There are times when a CICS region needs to be replicated, and then we must allocate and copy the application data with new dataset names. This process is long and tiresome – especially if you need to copy the data from every dataset in that region and then enter the CICS CSD file and change every file to point to the new related dataset.

This REXX routine provides an easy way to replicate a CICS data region by automatically generating back-up/restore commands for every dataset in CICS as input to DMS (from Sterling Software), and then automatically generating DEFINE command for a CICS batch mode run (except IBM internal files, eg DFH$...).

SYSTEM ENVIRONMENT

This process was used in the following environment:

- OS/390 Version 2.4
- CICS Version 4.1
- DMS Version 8.1.

STEP 1: CICS CSD LIST

First of all list the CSD with the following job:

```
//S038CSDL JOB (SS38,B1,30(,CSD-LIST,
//        NOTIFY=S038,MSGCLASS=T,REGION=5000K
//LISTCSD  EXEC PGM=DFHCSDUP
//STEPLIB  DD DSN=SYSP.CICS410.SDFHLOAD,DISP=SHR
//DFHCSD   DD DSN=SYSP.CICS410.WDFHCSD,DISP=SHR
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD DSN=S038.TEXT3,DISP=SHR
//SYSIN    DD *
   LIST  ALL OBJECTS
/*
//
```

STEP 2: AUTOMATIC COMMAND GENERATION

This is the main procedure and it has two purposes:

- Extract the FILE parameter from the CICS CSD (except IBM internal files).
- Generate DMS back-up/restore commands.

```
/* REXX ————————————————————— */
/*                                                    */
/* This REXX routine extracts the FILE parameters from   */
/* the CICS CSD file listed before with:              */
/*     'LIST  ALL OBJECTS'                            */
/* using DFHCSDUP.                                 */
/* The parms are inserted on the DEFINE FILE command     */
/* with new dataset names for CICS replication.          */
/*                                                    */
/* NOTE:                                              */
/* Before executing this REXX, change the 'NEW' parm      */
/* to the needed second-level qualifier.              */
/*                                                    */
/* ————————————————————————— */
TRACE N
ADDRESS TSO 'ALLOC FILE(IN) DA(SØ38.FCT.LIST) SHR'
ADDRESS TSO 'ALLOC FILE(OUT) DA(SØ38.FCT.DEF) SHR'
ADDRESS TSO 'ALLOC FILE(OUT1) DA(SØ38.FCT.BCK) SHR'
ADDRESS TSO 'ALLOC FILE(OUT2) DA(SØ38.FCT.RST) SHR'
I=Ø
N=Ø
NDS = Ø
NFI = Ø
NEW = 'NEW'
LOOP1:
"EXECIO 1 DISKR IN "
IF RC > Ø THEN SIGNAL OUT
  PULL ENTRY
  DUMMY = ENTRY
  DSN = ENTRY
  DESC = ENTRY
  LSR = ENTRY
  REM = ENTRY
  KEY = ENTRY
  STAT = ENTRY
  DATA = ENTRY
  TABLE = ENTRY
  DATAB = ENTRY
  RECFM = ENTRY
  ADD1 = ENTRY
  READ = ENTRY
  JOURNAL = ENTRY
  JULDATE= ENTRY
  RECOV = ENTRY
  STRING = ENTRY
```

```
KEYL = ENTRY
 PARSE VAR ENTRY FILE GROUP .
       IF SUBSTR(FILE,1,5) = 'FILE(' THEN DO
          IF SUBSTR(GROUP,7,4) = 'DFH$' THEN SIGNAL LOOP1
          IF SUBSTR(GROUP,7,7) = 'DFHMROF' THEN SIGNAL LOOP1
          IF SUBSTR(GROUP,7,7) = 'DFHCMAC' THEN SIGNAL LOOP1
                 FILE=SUBSTR(FILE,6,7)
                 FILE=STRIP(FILE,B,'(')
                 FILE=STRIP(FILE,B,' ')
                 FILE=STRIP(FILE,B,')')
                 GROUP=SUBSTR(GROUP,7,8)
                 GROUP=STRIP(GROUP,B,'(')
                 GROUP=STRIP(GROUP,B,' ')
                 GROUP=STRIP(GROUP,B,')')
                 FI = FILE
                 GR = GROUP
                 I=1              /* WRITE flag */
                 N = N + 1        /* LINE flag  */
                 NFI = NFI + 1
                 SIGNAL LOOP1
       END
 PARSE VAR DUMMY TEST .
       IF SUBSTR(TEST,1,15) = 'VSAM-PARAMETERS' THEN
          SIGNAL LOOP1
 PARSE VAR DESC DESCR .
       IF SUBSTR(DESCR,1,4) = 'DESC' THEN DO
                 TEMP = DESC
                 TEMP=SUBSTR(TEMP,53,8Ø)
                 TEMP=STRIP(TEMP,B,'(')
                 TEMP=STRIP(TEMP,B,')')
                 DES = TEMP
                 SIGNAL LOOP1
       END
 PARSE VAR DUMMY TEST .
       IF SUBSTR(TEST,1,15) = 'VSAM-PARAMETERS' THEN
          SIGNAL LOOP1
 PARSE VAR DSN DSN .
       IF SUBSTR(DSN,1,7) = 'DSNAME(' THEN DO
                 DSN=SUBSTR(DSN,8,44)
                 DSN=STRIP(DSN,B,' ')
                 DSN=STRIP(DSN,B,')')
                  OLDDSN = DSN
                  PARSE VAR DSN HLQ '.' REST .
                  DSN = HLQ || '.' || NEW || '.' || REST
                  NDS = NDS + 1
                 DS = DSN
                 SIGNAL LOOP1
       END
 PARSE VAR LSR PSWD LSR SHR .
       IF SUBSTR(PSWD,1,4) = 'PASS' THEN DO
```

```
                LSR=SUBSTR(LSR,1Ø,2)
                LSR=STRIP(LSR,B,' ')
                LSR=STRIP(LSR,B,')')
                LSR=STRIP(LSR,B,' ')
                LSR=STRIP(LSR,B,'(')
                PSWD=SUBSTR(PSWD,1Ø,8)
                PSWD=STRIP(PSWD,B,' ')
                PSWD=STRIP(PSWD,B,')')
                PSWD=STRIP(PSWD,B,'(')
                SHR=SUBSTR(SHR,12,8)
                SHR=STRIP(SHR,B,' ')
                SHR=STRIP(SHR,B,')')
                SHR=STRIP(SHR,B,' ')
                SHR=STRIP(SHR,B,'(')
                LS = LSR
                PS = PSWD
                SH = SHR
                SIGNAL LOOP1
        END
PARSE VAR STRING STRIN NSR1 .
        IF SUBSTR(STRIN,1,8) = 'STRINGS(' THEN DO
                STRIN=SUBSTR(STRIN,9,2)
                STRIN=STRIP(STRIN,B,')')
                STRIN=STRIP(STRIN,B,'(')
                NSR1=SUBSTR(NSR1,1Ø,8)
                NSR1=STRIP(NSR1,B,')')
                NSR1=STRIP(NSR1,B,'(')
                STR = STRIN
                NSR = NSR1
                SIGNAL LOOP1
        END
PARSE VAR DUMMY TEST .
        IF SUBSTR(TEST,1,17) = 'REMOTE-ATTRIBUTES' THEN
           SIGNAL LOOP1
PARSE VAR REM REMSYS REMT1 RECS .
        IF SUBSTR(RECS,1,8) = 'TRANSID(' THEN SIGNAL LOOP1
        IF SUBSTR(REMSYS,1,13) = 'REMOTESYSTEM(' THEN DO
                REMSYS=SUBSTR(REMSYS,14,8)
                REMSYS=STRIP(REMSYS,B,'(')
                REMSYS=STRIP(REMSYS,B,' ')
                REMSYS=STRIP(REMSYS,B,')')
                REMT1=SUBSTR(REMT1,12,8)
                REMT1=STRIP(REMT1,B,'(')
                REMT1=STRIP(REMT1,B,' ')
                REMT1=STRIP(REMT1,B,')')
                RECS=SUBSTR(RECS,12,5)
                RECS=STRIP(RECS,B,'(')
                RECS=STRIP(RECS,B,' ')
                RECS=STRIP(RECS,B,')')
                REMS = REMSYS
```

```
                  REMT = REMT1
                  REC = RECS
/*                SAY FI GR DES DS */
                  SIGNAL LOOP1
         END
   PARSE VAR KEYL KEY1 .
         IF SUBSTR(KEY1,1,1Ø) = 'KEYLENGTH(' THEN DO
                  KEY1=SUBSTR(KEY1,11,3)
                  KEY1=STRIP(KEY1,B,'(')
                  KEY1=STRIP(KEY1,B,' ')
                  KEY1=STRIP(KEY1,B,')')
                  KE = KEY1
                  SIGNAL LOOP1
         END
   PARSE VAR DUMMY TEST .
         IF SUBSTR(TEST,1,14) = 'INITIAL-STATUS' THEN
            SIGNAL LOOP1
   PARSE VAR STAT STA OPE DISP
         IF SUBSTR(STA,1,7) = 'STATUS(' THEN DO
                  STA=SUBSTR(STA,8,8)
                  STA=STRIP(STA,B,'(')
                  STA=STRIP(STA,B,' ')
                  STA=STRIP(STA,B,')')
                  ST = STA
                  STA = ' '
                  DISP=SUBSTR(DISP,13,8)
                  DISP=STRIP(DISP,B,'(')
                  DISP=STRIP(DISP,B,' ')
                  DISP=STRIP(DISP,B,')')
                  SIGNAL LOOP1
         END
   PARSE VAR DUMMY TEST .
         IF SUBSTR(TEST,1,7) = 'BUFFERS' THEN
            SIGNAL LOOP1
   PARSE VAR DATAB DATABUF IXBUF .
         IF SUBSTR(DATABUF,1,12) = 'DATABUFFERS(' THEN DO
                  DATABUF=SUBSTR(DATABUF,13,3)
                  DATABUF=STRIP(DATABUF,B,'(')
                  DATABUF=STRIP(DATABUF,B,' ')
                  DATABUF=STRIP(DATABUF,B,')')
                  IXBUF=SUBSTR(IXBUF,13,3)
                  IXBUF=STRIP(IXBUF,B,'(')
                  IXBUF=STRIP(IXBUF,B,' ')
                  IXBUF=STRIP(IXBUF,B,')')
                  DB = DATABUF
                  IXB = IXBUF
            SIGNAL LOOP1
         END
   PARSE VAR DUMMY TEST .
         IF SUBSTR(TEST,1,2Ø) = 'DATATABLE-PARAMETERS' THEN
```

```
         SIGNAL LOOP1
PARSE VAR TABLE TAB MAXRECS .
      IF SUBSTR(TAB,1,6) = 'TABLE(' THEN DO
                TAB=SUBSTR(TAB,7,3)
                TAB=STRIP(TAB,B,'(')
                TAB=STRIP(TAB,B,' ')
                TAB=STRIP(TAB,B,')')
                MAXRECS=SUBSTR(MAXRECS,12,3Ø)
                MAXRECS=STRIP(MAXRECS,B,'(')
                MAXRECS=STRIP(MAXRECS,B,' ')
                MAXRECS=STRIP(MAXRECS,B,')')
                TB = TAB
                MAXR = MAXRECS
         SIGNAL LOOP1
      END
PARSE VAR DUMMY TEST .
      IF SUBSTR(TEST,1,11) = 'DATA-FORMAT' THEN
         SIGNAL LOOP1
PARSE VAR RECFM RECF .
      IF SUBSTR(RECF,1,13) = 'RECORDFORMAT(' THEN DO
                RECF=SUBSTR(RECF,14,1)
                RECF=STRIP(RECF,B,'(')
                RECF=STRIP(RECF,B,' ')
                RECF=STRIP(RECF,B,')')
                RF = RECF
                SIGNAL LOOP1
      END
PARSE VAR DUMMY TEST .
      IF SUBSTR(TEST,1,1Ø) = 'OPERATIONS' THEN
         SIGNAL LOOP1
PARSE VAR ADD1 ADD BROWSE DELETE .
      IF SUBSTR(ADD,1,4) = 'ADD(' THEN DO
                ADD=SUBSTR(ADD,5,3)
                ADD=STRIP(ADD,B,'(')
                ADD=STRIP(ADD,B,' ')
                ADD=STRIP(ADD,B,')')
                BROWSE=SUBSTR(BROWSE,8,3)
                BROWSE=STRIP(BROWSE,B,'(')
                BROWSE=STRIP(BROWSE,B,' ')
                BROWSE=STRIP(BROWSE,B,')')
                DELETE=SUBSTR(DELETE,8,3)
                DELETE=STRIP(DELETE,B,'(')
                DELETE=STRIP(DELETE,B,' ')
                DELETE=STRIP(DELETE,B,')')
                AD = ADD
                BR = BROWSE
                DL = DELETE
                SIGNAL LOOP1
      END
PARSE VAR READ READ1 UPDATE .
```

```
      IF SUBSTR(READ1,1,5) = 'READ(' THEN DO
                READ1=SUBSTR(READ1,6,3)
                READ1=STRIP(READ1,B,'(')
                READ1=STRIP(READ1,B,' ')
                READ1=STRIP(READ1,B,')')
                UPDATE=SUBSTR(UPDATE,8,3)
                UPDATE=STRIP(UPDATE,B,'(')
                UPDATE=STRIP(UPDATE,B,' ')
                UPDATE=STRIP(UPDATE,B,')')
                RD = READ1
                UP = UPDATE
                READ1 = ' '
                UPDATE = ' '
                SIGNAL LOOP1
      END
PARSE VAR DUMMY TEST .
      IF SUBSTR(TEST,1,16) = 'AUTO-JOURNALLING' THEN
         SIGNAL LOOP1
PARSE VAR JOURNAL JRNL JNLR JNLS
      IF SUBSTR(JRNL,1,8) = 'JOURNAL(' THEN DO
                JRNL=SUBSTR(JRNL,9,3)
                JRNL=STRIP(JRNL,B,'(')
                JRNL=STRIP(JRNL,B,' ')
                JRNL=STRIP(JRNL,B,')')
                JNLR=SUBSTR(JNLR,9,1Ø)
                JNLR=STRIP(JNLR,B,'(')
                JNLR=STRIP(JNLR,B,' ')
                JNLR=STRIP(JNLR,B,')')
                JNLR=STRIP(JNLR,B,' ')
                JNLS=STRIP(JNLS,B,' ')
                JNLS=SUBSTR(JNLS,13,3)
                JNLS=STRIP(JNLS,B,'(')
                JNLS=STRIP(JNLS,B,' ')
                JNLS=STRIP(JNLS,B,')')
                JN = JRNL
                JR = JNLR
                JS = JNLS
                SIGNAL LOOP1
      END
PARSE VAR JULDATE JNLU JNLA JNLSW
      IF SUBSTR(JNLU,1,1Ø) = 'JNLUPDATE(' THEN DO
                JNLU=SUBSTR(JNLU,11,3)
                JNLU=STRIP(JNLU,B,'(')
                JNLU=STRIP(JNLU,B,' ')
                JNLU=STRIP(JNLU,B,')')
                JNLA=SUBSTR(JNLA,8,6)
                JNLA=STRIP(JNLA,B,'(')
                JNLA=STRIP(JNLA,B,' ')
                JNLA=STRIP(JNLA,B,')')
                JNLSW=STRIP(JNLSW,B,' ')
```

```
                  JNLSW=SUBSTR(JNLSW,14,3)
                  JNLSW=STRIP(JNLSW,B,'(')
                  JNLSW=STRIP(JNLSW,B,' ')
                  JNLSW=STRIP(JNLSW,B,')')
                  JU = JNLU
                  JA = JNLA
                  JW = JNLSW
                  SIGNAL LOOP1
         END
   PARSE VAR DUMMY TEST .
         IF SUBSTR(TEST,1,19) = 'RECOVERY-PARAMETERS' THEN
            SIGNAL LOOP1
   PARSE VAR RECOV RECV FWDR BACK .
         IF SUBSTR(RECV,1,9) = 'RECOVERY(' THEN DO
                  RECV=SUBSTR(RECV,1Ø,11)
                  RECV=STRIP(RECV,B,' ')
                  RECV=STRIP(RECV,B,'(')
                  RECV=STRIP(RECV,B,' ')
                  RECV=STRIP(RECV,B,')')
                  FWDR=SUBSTR(FWDR,13,2)
                  FWDR=STRIP(FWDR,B,' ')
                  FWDR=STRIP(FWDR,B,'(')
                  FWDR=STRIP(FWDR,B,' ')
                  FWDR=STRIP(FWDR,B,')')
                  BACK=SUBSTR(BACK,12,7)
                  BACK=STRIP(BACK,B,' ')
                  BACK=STRIP(BACK,B,'(')
                  BACK=STRIP(BACK,B,' ')
                  BACK=STRIP(BACK,B,')')
                  RCV = RECV
                  FW  = FWDR
                  BC  = BACK
                  IF I > Ø THEN SIGNAL WRITEDF
                  SIGNAL LOOP1
         END
CLEAR:
    FILE =
    DESC =
    GROUP =
    TEMP =
    DSN =
    LSR =
    PSWD =
    SHR =
    STRIN =
    NSR1 =
    REMSYS =
    REMT1 =
    RECS =
    KEY1 =
```

```
    RECF =
    ADD =
    BROWSE =
    DELETE =
    JRNL =
    JNLR =
    JNLS =
    JNLU =
    JNLA =
    JNLSW =
    RECV =
    FWDR =
    BACK =
  SIGNAL LOOP1
WRITEDF:
OUT. =
OUT.N =  ' DEFINE FILE(' || FI || ')  GROUP(' || GR || ')'
IF DS ¬=  '' THEN DO
   N = N+1
   OUT.N =  '        DSNAME(' || DS || ')'
   SIGNAL S1
END
S1:
IF DES ¬= '' THEN do
   N = N+1
   OUT.N =  '        DESCRIPTION(' || DES || ')'
   SIGNAL S2
end
S2:
IF STRIP(LS,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        LSRPOOLID(' || LS || ')'
   SIGNAL S3
END
S3:
IF STRIP(SH,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        DSNSHARING(' || SH || ')'
   SIGNAL S4
END
S4:
IF STRIP(STR,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        STRINGS(' || STR || ')'
   SIGNAL S5
END
S5:
IF STRIP(NSR,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        NSRGROUP(' || NSR || ')'
```

```
   SIGNAL S6
END
S6:
IF STRIP(REMS,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        REMOTESYSTEM(' || REMS || ')'
   SIGNAL S7
END
S7:
IF STRIP(REMT,B,' ') ¬= '' THEN DO
   N = N + 1
   OUT.N =  '        REMOTENAME(' || REMT || ')'
   SIGNAL S8
END
S8:
IF STRIP(REC,B,' ') ¬= '' THEN DO
   N = N + 1
   OUT.N =   '        RECORDSIZE(' || REC || ')'
   SIGNAL S9
END
S9:
IF STRIP(KE,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =   '        KEYLENGTH(' || KE || ')'
   SIGNAL S1Ø
END
S1Ø:
IF STRIP(ST,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        STATUS(' || ST || ')'
   SIGNAL S11
END
S11:
IF STRIP(RF,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        RECORDFORMAT(' || RF || ')'
   SIGNAL S12
END
S12:
IF STRIP(AD,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        ADD(' || AD || ') BROWSE(' || BR || ,
        ') DELETE(' || DL || ') READ(' || RD || ') UPDATE(' || UP || ')'
   SIGNAL S13
END
S13:
IF STRIP(JN,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        JOURNAL(' || JN || ')'
   SIGNAL S14
```

```
END
S14:
IF STRIP(JR,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        JNLREAD (' || JR || ')'
   SIGNAL S15
END
S15:
IF STRIP(JS,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        JNLSYNCREAD(' || JS || ')'
   SIGNAL S16
END
S16:
IF STRIP(JU,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        JNLUPDATE(' || JU || ')'
   SIGNAL S17
END
S17:
IF STRIP(JA,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        JNLADD(' || JA || ')'
   SIGNAL S18
END
S17:
IF STRIP(JA,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        JNLSYNCWRITE(' || JW || ')'
   SIGNAL S18
END
S18:
IF STRIP(RCV,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        RECOVERY(' || RCV || ') '
   SIGNAL S19
END
S19:
IF STRIP(FW,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        FWDRECOVLOG(' || FW || ')'
   SIGNAL S2Ø
END
S2Ø:
IF STRIP(BC,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        BACKUPTYPE(' || BC || ')'
   SIGNAL S21
END
```

```
S21:
IF STRIP(DB,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        DATABUFFERS(' || DB || ')'
   SIGNAL S22
END
S22:
IF STRIP(IXB,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        INDEXBUFFERS(' || IXB || ')'
   SIGNAL S23
END
S23:
IF STRIP(TB,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        TABLE(' || TB || ')'
   SIGNAL S24
END
S24:
IF STRIP(MAXR,B,' ') ¬= '' THEN DO
   N = N+1
   OUT.N =  '        MAXNUMRECS(' || MAXR || ')'
   SIGNAL WRITEDEF
END
WRITEDEF:
  "EXECIO * DISKW OUT (STEM OUT."
  IF RC > 1 THEN DO
            SAY 'ERROR WRITING ON DATASET :'
            SIGNAL OUT
            END
WRITEBCK:
OUT1. =
OUT1.1 = 'FIND DSN=' || OLDDSN
OUT1.2 = '     BACKUP,RETPD=30'
  "EXECIO * DISKW OUT1 (STEM OUT1."
  IF RC > 1 THEN DO
            SAY 'ERROR WRITING ON DATASET :'
            SIGNAL OUT
            END
WRITERST:
OUT2. =
OUT2.1 =  ' RESTORE DSN=' || OLDDSN
OUT2.2 =  '         NEW=' || DS || ',VOL=volser,REPLACE'
  "EXECIO * DISKW OUT2 (STEM OUT2."
  IF RC > 1 THEN DO
            SIGNAL OUT
            SAY 'ERROR WRITING ON DATASET :'
            END
N=0
```

```
I=Ø
SIGNAL LOOP1
OUT:
  SAY ' YOU HAVE = ' || NFI || ' FILES AND ' || NDS || ' DS'
  "EXECIO Ø DISKR IN (FINIS"
  ADDRESS TSO "FREE F(IN)"
  "EXECIO Ø DISKW OUT (FINIS"
  ADDRESS TSO "FREE F(OUT)"
  "EXECIO Ø DISKW OUT1 (FINIS"
  ADDRESS TSO "FREE F(OUT1)"
  "EXECIO Ø DISKW OUT2 (FINIS"
  ADDRESS TSO "FREE F(OUT2)"
  EXIT
```

DEFINE FILE FOR CICS CSD

Note: the parameters for the DEFINE command generated depend on the source FILE (ie nulls or blank records are excluded and not coded in the command).

```
DEFINE FILE(DLPKCHK)  GROUP(DLPKFCT)
       DSNAME(ABCD.NEW.STEST.AV.DLPKCHK.VSAMC)
       DESCRIPTION(REPLACEMENT FOR CMXT CLASS 1Ø)
       LSRPOOLID(1)
       DSNSHARING(ALLREQS)
       STRINGS(1)
       STATUS(ENABLED)
       RECORDFORMAT(F)
       ADD(YES) BROWSE(YES) DELETE(YES) READ(YES) UPDATE(YES)
       JOURNAL(NO)
       JNLREAD (NONE)
       JNLSYNCREAD(NO)
       JNLUPDATE(NO)
       JNLADD(NONE)
       RECOVERY(NONE)
       FWDRECOVLOG(NO)
       BACKUPTYPE(STATIC)
       DATABUFFERS(2)
       INDEXBUFFERS(1)
       TABLE(NO)
```

DMS BACK-UP RESTORE COMMANDS

Back-up commands

```
FIND DSN=ABCD.STEST.AV.DLPKCHK.VSAMC
```

```
     BACKUP,RETPD=3Ø
FIND DSN=ABCD.STSET.TNU.DLPKLOG.VSAMC
     BACKUP,RETPD=3Ø
FIND DSN=ABCD.STEST.DLPKTAM.T25Ø13.VSAM
     BACKUP,RETPD=30
```

Restore commands

Note: you should change the VOLSER to the target needed.

```
RESTORE DSN=ABCD.STEST.AV.DLPKCHK.VSAMC
         NEW=ABCD.NEW.STEST.AV.DLPKCHK.VSAMC,VOL=volser,REPLACE
 RESTORE DSN=ABCD.STSET.TNU.DLPKLOG.VSAMC
         NEW=ABCD.NEW.STSET.TNU.DLPKLOG.VSAMC,VOL=volser,REPLACE
 RESTORE DSN=ABCD.STEST.DLPKTAM.T25Ø13.VSAM
       NEW=ABCD.NEW.STEST.DLPKTAM.T25013.VSAM,VOL=volser,REPLACE
```

DMS back-up/restore JCL

```
//SØ38DMSB JOB (SS38,B1,3Ø),CSD-SELECTIVE-BACKUP,
//        NOTIFY=SØ38,MSGCLASS=T,REGION=5ØØØK
//*
//* *——————————————————————————*
//* * BACK-UP OF SELECTED DS FOR        *
//* * FILES IN CICS CSD                 *
//* *                                   *
//* *——————————————————————————*
//*
//S1       EXEC  DMS
//SYSIN  DD DSN=SØ38.FCT.BCK,DISP=SHR for Backup
//*SYSIN DD DSN=SØ38.FCT.RST,DISP=SHR for Restore
```

CICS batch definition for CSD

```
//SØ38CSDB JOB (SS38,B1,3Ø),CSD-BATCH-DEFINE,
//        NOTIFY=SØ38,MSGCLASS=T,REGION=5ØØØK
//LISTCSD  EXEC PGM=DFHCSDUP
//STEPLIB  DD DSN=SYSP.CICS41Ø.SDFHLOAD,DISP=SHR
//DFHCSD   DD DSN=SYSP.CICS41Ø.ZDFHCSD,DISP=SHR
//SYSOUT   DD SYSOUT=*
//SYSPRINT DD SYSOUT=*
//SYSIN    DD DSN=SO38.FCT.DEF,DISP=SHR
```

Oded Tagger
Senior Systems Programmer (Israel)

Administering RDO resources in the DFHCSD

Many installations are using the CICS system definition utility program DFHCSDUP to administer the CSD file in batch mode. To provide a more convenient interface for our CICS administrators, I have written the RDODEF EXEC (one REXX EXEC and two ISPF panels), which uses the program interface to DFHCSDUP to maintain the RDO resources interactively under TSO/ISPF.

Because the DFHCSDUP load module is called as an external routine, the CICS SDFHLOAD library must be included in the TSO STEPLIB concatenation.

Initially, a selection panel (RDODEF1) asks for the RDO source member that is to be modified. These source members contain the RDO commands for the associated RDO group or list. After receiving the member name, an edit session is entered.

Panel RDODEF2 then prompts for the execution of the commands. The required datasets for utility invocation are allocated and DFHCSDUP is called.

If the return code for the call command signals successful processing, SYSPRINT is displayed to verify the DFHCSDUP execution. Returning (PF3-End) to the selection screen (RDODEF1) will free all datasets and also delete the temporary dataset related to SYSPRINT. CICS administrators can then use the CEDA INSTALL transaction to install the previously maintained RDO resource group in the CICS region.

RDODEF EXEC

```
/* REXX */
/*====================================================================-/
/- Function: Administer RDO resources.                                -/
/-           The user will be prompted before the RDO command(s)     -/
/-           are executed by DFHCSD.                                 -/
/-====================================================================-*/
csddsn   = 'CICSESA.DFHCSD'                  /* DFHCSD dataset name */
lib      = 'CICSESA.RDODEFS'                 /* RDO source library  */
```

```
lib2     = 'CICSESA.SDFHLOAD'                  /* utility library     */
rdopanl1 = 'RDODEF1'                           /* selection panel     */
rdopanl2 = 'RDODEF2'                           /* defstep prompt panel*/
rdomem   = ''                                  /* init panel fields   */
msg      = ''
do forever
    ADDRESS ISPEXEC 'DISPLAY PANEL (' rdopanl1 ')'
    if rc = 8 then
       leave                       /* pf3 - end                       */
    else
     do
       x = outtrap(var.,'*')
       member = lib || '(' || rdomem || ')'      /* search for RDO    */
       ADDRESS TSO "LISTDS ('"member"')"         /* source member     */
       if rc = 0 then
          do
            ADDRESS ISPEXEC "EDIT DATASET('"member"')"
            defmsg = 'Source edit step executed!'   /* init def step  */
            defstep = 'Y'                          /* prompt panel      */
            ADDRESS ISPEXEC 'ADDPOP'
            ADDRESS ISPEXEC 'DISPLAY PANEL ('rdopanl2 ')'
            ADDRESS ISPEXEC 'REMPOP'
            if defstep = 'Y' then                  /* execute define ?  */
              do
                alloc_rc = 0
                call alloc                         /* alloc temp ds     */
                if alloc_rc = 0 then
                 do
                   call defstep                    /* run utility       */
                   if  defstep_rc <= 4 then        /* define executed   */
                     do                            /* display SYSPRINT  */
                       ADDRESS ISPEXEC "VIEW DATASET('"dsntemp2"')"
                       call dealloc                /* del temp ds       */
                       msg = 'Define step executed!'
                     end
                   else
                     do
                       call dealloc                /* del temp ds       */
                       msg = 'Define step unsuccessful!'
                     end
                 end
                else   call dealloc                /* del temp ds       */
              end
            else
              do
                msg = 'Define step not executed!'
              end
          end
       else
          do
```

```
            msg = 'RDO group not found!'
          end
     end
end
exit
/*--------------------------------------------------------------------/
/- subroutine: alloc temp datasets for utility execution             -/
/-             dsntemp1 = SYSIN                                      -/
/-             dsntemp2 = SYSPRINT                                   -/
/-             dsntemp3 = DFHCSD                                     -/
/--------------------------------------------------------------------*/
alloc:
temp1 = 'SYSIN'
temp2 = 'SYSPRINT'
temp3 = 'DFHCSD'
dsntemp1 = member
dsntemp2 = USERID() || '.RDODEF' || '.SYSPRINT'
dsntemp3 = csddsn
/*                                          allocate temp dataset 1    */
ADDRESS TSO "ALLOC FI("temp1") DA('"dsntemp1"') SHR"
if rc ¬= Ø then
  do
    tsoemsg = MSG("OFF")
    dsnstat = SYSDSN("'"dsntemp1"'")
    tsoemsg = MSG(tsoemsg)
    msg = 'SYSIN cannot be allocated!' dsnstat
    alloc_rc = 1
    return
  end
/*                                          allocate temp dataset 2    */
ADDRESS TSO "ALLOC FI("temp2") DA('"dsntemp2"') OLD REUSE"
if rc = Ø then
   ADDRESS TSO "EXECIO Ø DISKW "temp2" (OPEN FINIS"
else
   ADDRESS TSO "ALLOC FI("temp2") DA('"dsntemp2"'),
   NEW CAT REUSE UNIT(SYSTS),
   LRECL(133) BLKSIZE(2793Ø) RECFM(F B) SPACE(1,1) CYLINDERS"
   if rc ¬= Ø then
      do
        tsoemsg = MSG("OFF")
        dsnstat = SYSDSN("'"dsntemp2"'")
        tsoemsg = MSG(tsoemsg)
        msg = 'SYSPRINT cannot be allocated!' dsnstat
        alloc_rc = 1
        return
      end
/*                                          allocate temp dataset 3    */
ADDRESS TSO "ALLOC FI("temp3") DA('"dsntemp3"') SHR"
if rc ¬= Ø then
   do
```

```
     tsoemsg = MSG("OFF")
     dsnstat = SYSDSN("'"dsntemp3"'")
     tsoemsg = MSG(tsoemsg)
     msg = 'DFHCSD cannot be allocated |' dsnstat
     alloc_rc = 1
     return
   end
return
/*--------------------------------------------------------------------*/
/- subroutine: dealloc and delete temp data sets                      -/
/--------------------------------------------------------------------*/
dealloc:
ADDRESS TSO "FREE  FI("temp1")"
ADDRESS TSO "FREE  FI("temp2")"
ADDRESS TSO "FREE  FI("temp3")"
ADDRESS TSO "DELETE ('"dsntemp2"')"
return
/*--------------------------------------------------------------------*/
/- subroutine: define RDO source to the CSD                           -/
/--------------------------------------------------------------------*/
defstep:
ADDRESS TSO "call '" || lib2 || "(DFHCSDUP)'"
defstep_rc = rc                              /* save return code    */
return
```

PANEL RDODEF1

```
)Body
%               ADMINISTER RDO RESOURCES                User   -+&ZUSER
%                                                       Date   -+&date
% COMMAND ===> _ZCMD                                   +%Time   -+&ZTIME
%─────────────────────────────────────
+
+               RSOURCE1 - Description
+               RSOURCE2 - Description
+               RSOURCE3 - Description
+               RSOURCE4 - Description
+               RSOURCE5 - Description
+
+              %in RDO source member:+ _RDOMEM
+
+
+
+               &MSG
%─────────────────────────────────────
%
%
)INIT
   .CURSOR = RDOMEM
```

```
   &DATE    = ,&ZDAY..&ZMONTH..&ZSTDYEAR'
)PROC
   VER (&rdomem,nb,len,'<=',8)
)END
```

PANEL RDODEF2

```
)Body  window(65,1Ø)
%COMMAND ===>_ZCMD
%———————————————————————————————
+
+              &defmsg
+
+             %Execute RDO commands to the DFHCSD:+ _defstep  +
+
%———————————————————————————————
)INIT
 .CURSOR = ZCMD
)REINIT
 REFRESH(*)
)PROC
   VER (&defstep,list,Y,N)
)END
```

Raimund Kleebaur
CICS Systems programmer
Hugo Boss AG (Germany)

Controlling CICS resources from a batch program using the CICSPlex SM API

We have a parallel sysplex environment consisting of two mainframes running OS/390 Version 2.6, two coupling facilities, and a variety of other components. Our CICS environment consists of one TOR connected to ten AORs, all of which are running Transaction Server 1.3. These regions have limited availability, from about 6am to 1am. We are in the process of setting up a second TOR and two new AORs to create a smaller CICSPlex, which will be available 24 hours a day, seven days a week (or close to it). The two TORs and two AORs will be split across the two mainframes so we can have one mainframe

down but still provide CICS availability on the other. We are using CICSPlex SM (CPSM) Version 1.4 to help us manage the CICS environment.

We have many application batch jobs that run while the CICS regions are still active. These jobs update VSAM files, which are defined to CICS, so it is necessary for us to disable transactions and close files before the jobs run. We previously used McKinney's CICS/CEMT product for this purpose; however, with our changing environment, we would have been forced to make many modifications to our production JCL to account for the CICS changes.

So to keep our environment changes from affecting our production JCL, and make our batch processes more sysplex friendly, I wrote a batch COBOL II program, which uses the CICSPlex SM API to control the resources. All of our production regions across the sysplex are defined to the same CPSM CICSGRP, so we can simply tell the batch program to close a file in the CICSGRP and it will close it on all of our production CICS regions no matter where the file is defined. This flexibility will make it easier for us to move applications to the new 24x7 regions as the applications become sysplex enabled.

At this point, the program will perform the following functions: CLOSE FILE, ENABLE FILE, OPEN FILE, DISABLE TRAN, ENABLE TRAN, DISABLE PROG, DISCARD PROG, ENABLE PROG, NEWCOPY PROG. I suspect that, as we get more experience with CPSM and our new environment, we'll expand the program's capabilities. The program performs a variety of edits and creates an error/status report. The program also ends normally under certain circumstances, such as the CICS regions being unavailable, so that the system contact for the application doesn't get a call if the batch process gets pushed back after the CICS regions come down.

Given below is a copy of the instructions that were given to our JCL area when they converted our production JCL from McKinney to the new process. Your CONTEXT and SCOPE names may, of course, vary.

SET-UP INSTRUCTIONS

These instructions describe how to set up JCL to use the SYS182

program to control CICS resources (transactions, files, and programs) from the batch environment. This process replaces McKinney's CICS/CEMT process.

Here is the sample JCL from CICS.PROD.CPSM.BATCH.CMDS ($SAMPJCL) you can use to start with:

```
//SYS182PØ  EXEC PGM=SYS182,REGION=2M
//STEPLIB   DD DSN=SYS.LINK.LIB,DISP=SHR
//          DD DSN=CICS.PROD.CPSM.SEYUAUTH,DISP=SHR
//ISEQLØ1   DD DISP=SHR,DSN=CICS.PROD.CPSM.BATCH.CMDS(xxxxxxxx)
//ISEQLØ2   DD DISP=SHR,DSN=CICS.PROD.CPSM.BATCH.CMDS($RSCGRPS)
//PRINTØ1   DD SYSOUT=Beta92 - accessible to JCL & CICS areas
//SYSOUT    DD SYSOUT=*
//SYSLST    DD SYSOUT=*
//SYSUDUMP  DD SYSOUT=*
```

ISEQL01 SET UP

The *xxxxxxxx* member specified for ISEQL01 contains the commands, or actions, that you want to carry out and the resources and environment on which to carry them out. The format of the ISEQL01 member must be as follows, with all text starting in column 1:

```
TEST (optional - for testing purposes)
CONTEXT=XXXXXXXX SCOPE=XXXXXXXX
ACTION=XXXXXXXXXXXXX
ONE OR MORE RESOURCE RECORDS
ACTION=XXXXXXXXXXXXX (opt)
ONE OR MORE RESOURCE RECORDS (opt)
ACTION=XXXXXXXXXXXXX (opt)
ONE OR MORE RESOURCE RECORDS (opt)
CONTEXT=XXXXXXXX SCOPE=XXXXXXXX (opt)
ACTION=XXXXXXXXXXXXX (opt)
ONE OR MORE RESOURCE RECORDS (opt)
```

'TEST' is optional, and should be coded only when you want to test the contents of ISEQL01 for accuracy. If TEST is coded and SYS182 is run, the program will verify:

- The context and scope are valid and can be connected to.
- The action is valid.
- At least one resource record is coded for the given action (it does not determine whether the resource(s) is valid, however).

- Any resource groups, if specified, can be found in ISEQL02.

The context and scope together describe the environment in which you want to operate. Valid values are described below.

CICSPLXP

- CICSPTOR = all production TOR regions – use this when disabling and enabling transactions.
- CICSPLXP = all production regions – use this instead of one of the other production options whenever possible!
- CICSPAOR = all production AOR regions.
- CICSP1 = all production regions on GSYS1.
- CICSP4 = all production regions on GSYS4.
- ACICICPE (or other specific production region APPLID).

CICSPLXM

- CICSPLXM = all demo regions – use this instead of one of the other demo options whenever possible.
- CICSM1 = all demo regions on GSYS1.
- CICSM4 = all demo regions on GSYS4.
- CICSTB (or other specific demo region APPLID).

CICSPLXD

- CICSPLXD = all development regions – use this instead of one of the other development options whenever possible.
- CICSD1 = all development regions on GSYS1.
- CICSD4 = all development regions on GSYS4.
- CICSDVLP (or other specific development region APPLID).

CICSPLXT

- CICSPLXT = all test regions – use this instead of one of the other test options whenever possible.
- CICSTT = all test regions on GSYS1.
- CICSTD = all test regions on GSYS4.
- CICSTGT (or other specific test region APPLID).

Actions

Valid actions are:

- CLOSE FILE – close and disable a file, making it unusable to CICS.
- ENABLE FILE – enable a file, leaving it closed. Doing this will be faster than opening the file. The file will be opened by CICS the first time someone tries to access it through an on-line transaction.
- OPEN FILE – enable and open a file.
- DISABLE TRAN – disable a transaction, making it unusable.
- ENABLE TRAN – enable a transaction.
- DISABLE PROG – disable a program, making it unusable.
- DISCARD PROG – discard a program from memory. The program must be disabled first. The next time someone tries to run the program, CICS will load a new version of it into memory from the applicable library (cics.userlib, cics.demolib, etc).
- ENABLE PROG – enable a program.
- NEWCOPY PROG – copy a new version of a program into memory from the applicable library (cics.userlib, cics.demolib, etc).

Resources

Valid resources are:

- Files – the (up to) 8-character DDname of the file on which to perform the given action. Alternatively, when you need to perform an action on a large number of resources in more than one job, you can instead specify a group name consisting of '@' followed by up to eight letters and/or numbers. The group name must match that of a name listed in member $RSCGRPS of the PDS CICS.PROD.CPSM.BATCH.CMDS. The action does not have to be the same in both jobs. This allows you, for example, to perform 'close file' on a group of resources in one job and 'open file' on the same group in another job. If more than one group is specified, they must be in the same order that they occur in $RSCGRPS. See the notes below regarding ISEQL02 for additional information.
- Transactions – the 4-character name of the transaction on which you want to perform the given action. You can instead specify a group name consisting of '@' followed by up to eight letters and/or numbers. See the notes below regarding ISEQL02 for additional information.
- Programs – the 8-character name of the program on which you want to perform the given action. You can instead specify a group name consisting of '@' followed by up to eight letters and/or numbers. See the notes below regarding ISEQL02 for additional information.

ISEQL02 SET UP

CICS.PROD.CPSM.BATCH.CMDS($RSCGRPS) contains groups of resources that are referenced in more than one job. A group consists of a group name record followed by two or more specific resource records. The group names consist of an '@' followed by up to eight letters and/or numbers (eg @dmygrp01, @gr123). The groups should be inserted in alphabetical order by group name. Letters come before numbers in the sort sequence (eg @group should be before @groupz, which should be before @group1). The order of the resources within the groups is not important to the functionality of the program, but they should be organized by name within the various groups to simplify future maintenance.

Here is an example for ISEQL01:

```
CONTEXT=CICSPLXP SCOPE=CICSPTOR
ACTION=DISABLE TRAN
TRN1
TRN2
CONTEXT=CICSPLXP SCOPE=CICSPLXP
ACTION=CLOSE FILE
FILEA1
FILEØ2
FILEØ1
ACTION=DISABLE PROG
PROGRAM1
```

Here is an example for ISEQL02:

```
@COMW1Ø
NCSCOMTR
@ESPDD
ANNALPH
ANNINFO
SPCADD
```

SYS182 COBOL PROGRAM

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.    SYS182.
      *****************************************************************
      *  AUTHOR:      RANDY SCHULTING.
      *  DESCRIPTION: THIS PROGRAM WAS WRITTEN TO REPLACE MCKINNEY
      *               SYSTEMS' "MTPBATCH" PROCESS. IT WILL PERFORM
      *               THE ACTIONS LISTED BELOW, ALL FROM THE BATCH
      *               ENVIRONMENT. IT DOES THIS USING CPSM, SO IT CAN
      *               PROCESS RESOURCES ON MULTIPLE CICS REGIONS AT THE
      *               SAME TIME.
      *
      *               VALID ACTIONS:
      *                  CLOSE FILE
      *                  ENABLE FILE
      *                  OPEN FILE
      *                  DISABLE TRAN
      *                  ENABLE TRAN
      *                  DISABLE PROG
      *                  DISCARD PROG
      *                  ENABLE PROG
      *                  NEWCOPY PROG
      *
      *       ****    THE FORMAT OF THE ISEQLØ1 FILE CAN BE TESTED BY
      *       ****    CODING THE WORD 'TEST' IN COLUMN 1 OF THE FIRST
```

```
*       ****    RECORD OF ISEQLØ1. IT SHOULD BE THE ONLY TEXT
*       ****    ON THE RECORD.
*
*               THE FORMAT OF ISEQLØ1 MUST BE AS FOLLOWS, WITH
*               ALL TEXT STARTING IN COLUMN 1. THE BLANK LINES
*               INSERTED BELOW ARE FOR READABILITY PURPOSES AND
*               SHOULD NOT BE CODED IN ISEQLØ1.
*
*               - TEST (OPT)
*               - CONTEXT=XXXXXXXX SCOPE=XXXXXXXX
*               - ACTION=XXXXXXXXXXXXX
*               - ONE OR MORE RESOURCE RECORDS
*               - ACTION=XXXXXXXXXXXXX (OPT)
*               - ONE OR MORE RESOURCE RECORDS (OPT)
*               - ACTION=XXXXXXXXXXXXX (OPT)
*               - ONE OR MORE RESOURCE RECORDS (OPT)
*
*               - CONTEXT=XXXXXXXX SCOPE=XXXXXXXX (OPT)
*               - ACTION=XXXXXXXXXXXXX (OPT)
*               - ONE OR MORE RESOURCE RECORDS (OPT)
*
****************************************************************

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT COMMAND-FILE-IN  ASSIGN TO UT-S-ISEQLØ1.
     SELECT GROUP-FILE-IN    ASSIGN TO UT-S-ISEQLØ2.
     SELECT PRINT-FILE-OUT   ASSIGN TO UT-S-PRINTØ1.

 DATA DIVISION.
 FILE SECTION.

 FD  COMMAND-FILE-IN
     LABEL RECORDS ARE STANDARD
     RECORDING MODE IS F
     BLOCK CONTAINS Ø RECORDS.
 Ø1  COMMAND-RECORD-IN       PIC X(8Ø).

 FD  GROUP-FILE-IN
     LABEL RECORDS ARE STANDARD
     RECORDING MODE IS F
     BLOCK CONTAINS Ø RECORDS.
 Ø1  GROUP-RECORD-IN         PIC X(8Ø).

 FD  PRINT-FILE-OUT
     LABEL RECORDS ARE STANDARD
     RECORDING MODE IS F
     BLOCK CONTAINS Ø RECORDS.
 Ø1  PRINT-LINE-OUT          PIC X(133).
```

```
WORKING-STORAGE SECTION.

Ø1  CRS-COMMAND-RECORD.
    Ø5  CRS-CONTEXT-SCOPE-RECORD.
        1Ø  CRS-CONTEXT-LIT      PIC X(8).
        1Ø  CRS-CONTEXT-SCOPE    PIC X(72).
    Ø5  FILLER REDEFINES CRS-CONTEXT-SCOPE-RECORD.
        1Ø  CRS-ACTION-LIT       PIC X(7).
        1Ø  CRS-ACTION           PIC X(12).
        1Ø  FILLER               PIC X(61).
    Ø5  FILLER REDEFINES CRS-CONTEXT-SCOPE-RECORD.
        1Ø  CRS-GROUP-IND        PIC X(1).
        1Ø  CRS-GROUP-NAME       PIC X(8).
        1Ø  FILLER               PIC X(71).
    Ø5  FILLER REDEFINES CRS-CONTEXT-SCOPE-RECORD.
        1Ø  CRS-RSRC-NAME        PIC X(8).
        1Ø  FILLER               PIC X(72).

Ø1  GRS-GROUP-RECORD.
    Ø5  GRS-GROUP-NAME-RECORD.
        1Ø  GRS-GROUP-IND       PIC X(1).
        1Ø  GRS-GROUP-NAME      PIC X(8).
        1Ø  FILLER              PIC X(71).
    Ø5  FILLER REDEFINES GRS-GROUP-NAME-RECORD.
        1Ø  GRS-GROUP-RSRC      PIC X(8).
        1Ø  FILLER              PIC X(72).

Ø1  HEAD-LINE1.
    Ø5  FILLER              PIC X(14)   VALUE ' SYS182 '.
    Ø5  FILLER              PIC X(42)   VALUE SPACES.
    Ø5  FILLER              PIC X(22)   VALUE
        'C. I. C. S.    A R E A'.
    Ø5  FILLER              PIC X(45)   VALUE SPACES.
    Ø5  H1-MM               PIC X(2)    VALUE SPACES.
    Ø5  FILLER              PIC X(1)    VALUE '/'.
    Ø5  H1-DD               PIC X(2)    VALUE SPACES.
    Ø5  FILLER              PIC X(1)    VALUE '/'.
    Ø5  H1-CCYY             PIC X(4)    VALUE SPACES.

Ø1  HEAD-LINE2.
    Ø5  FILLER              PIC X(53)   VALUE SPACES.
    Ø5  FILLER              PIC X(27)   VALUE
        'CPSM ERROR & SUMMARY REPORT'.
    Ø5  FILLER              PIC X(53)   VALUE SPACES.

Ø1  EL1-ERROR-LINE1.
    Ø5  FILLER              PIC X       VALUE SPACE.
    Ø5  EL1-ERROR-MSG       PIC X(132)  VALUE SPACES.

Ø1  EL2-ERROR-LINE2.
```

```
    Ø5  FILLER              PIC X(5)    VALUE SPACES.
    Ø5  FILLER              PIC X(1Ø)   VALUE 'RESPONSE: '.
    Ø5  EL2-RESPONSE        PIC X(12)   VALUE SPACES.
    Ø5  FILLER              PIC X(1Ø)   VALUE '  REASON: '.
    Ø5  EL2-REASON          PIC X(12)   VALUE SPACES.
    Ø5  FILLER              PIC X(11)   VALUE '  CONTEXT: '.
    Ø5  EL2-CONTEXT         PIC X(8)    VALUE SPACES.
    Ø5  FILLER              PIC X(9)    VALUE '  SCOPE: '.
    Ø5  EL2-SCOPE           PIC X(8)    VALUE SPACES.
    Ø5  FILLER              PIC X(1Ø)   VALUE '  ACTION: '.
    Ø5  EL2-ACTION          PIC X(8)    VALUE SPACES.
    Ø5  FILLER              PIC X(9)    VALUE '  RSRCE: '.
    Ø5  EL2-RESOURCE        PIC X(21)   VALUE SPACES.

Ø1  ML1-MESSAGE-LINE1.
    Ø5  FILLER              PIC X(5)    VALUE SPACES.
    Ø5  ML1-MESSAGE         PIC X(44)   VALUE SPACES.
    Ø5  FILLER              PIC X(11)   VALUE '  CONTEXT: '.
    Ø5  ML1-CONTEXT         PIC X(8)    VALUE SPACES.
    Ø5  FILLER              PIC X(9)    VALUE '  SCOPE: '.
    Ø5  ML1-SCOPE           PIC X(8)    VALUE SPACES.
    Ø5  FILLER              PIC X(1Ø)   VALUE '  ACTION: '.
    Ø5  ML1-ACTION          PIC X(8)    VALUE SPACES.
    Ø5  FILLER              PIC X(9)    VALUE '  RSRCE: '.
    Ø5  ML1-RESOURCE        PIC X(21)   VALUE SPACES.

Ø1  NO-CONTEXT-MSG.
    Ø5  FILLER              PIC X       VALUE SPACE.
    Ø5  FILLER              PIC X(52)   VALUE
        '*** INVALID RECORD FORMAT RECEIVED FOR CONTEXT/SCOPE'.
    Ø5  FILLER              PIC X(35)   VALUE
        ' IN ISEQLØ1. SHOULD BE (COLUMN 1): '.
    Ø5  FILLER              PIC X(43)   VALUE
        'CONTEXT=XXXXXXXX SCOPE=XXXXXXXX   RECEIVED:'.
    Ø5  FILLER              PIC X(2)    VALUE SPACES.

Ø1  INVALID-CONTEXT-MSG.
    Ø5  FILLER              PIC X       VALUE SPACE.
    Ø5  FILLER              PIC X(51)   VALUE
        '*** INVALID CONTEXT RECEIVED IN ISEQLØ1. SHOULD BE '.
    Ø5  FILLER              PIC X(45)   VALUE
        'CICSPLXP-PROD, CICSPLXM-DEMO, CICSPLXD-DVLP, '.
    Ø5  FILLER              PIC X(36)   VALUE
        'OR CICSPLXT-TEST.   RECEIVED:       '.

Ø1  NO-SCOPE-MSG.
    Ø5  FILLER              PIC X       VALUE SPACE.
    Ø5  FILLER              PIC X(49)   VALUE
        '*** SCOPE NOT FOUND ON RECORD IN ISEQLØ1. FORMAT '.
    Ø5  FILLER              PIC X(45)   VALUE
        'SHOULD BE (COLUMN 1): CONTEXT=XXXXXXXX SCOPE='.
```

```
    05  FILLER              PIC X(20)   VALUE
        'XXXXXXXX   RECEIVED:'.
    05  FILLER              PIC X(18)   VALUE SPACES.

01  INVALID-SCOPE-MSG.
    05  FILLER              PIC X       VALUE SPACE.
    05  FILLER              PIC X(46)   VALUE
        '*** INVALID SCOPE RECEIVED FOR THE CONTEXT IN '.
    05  FILLER              PIC X(20)   VALUE
        'ISEQL01.   RECEIVED:'.
    05  FILLER              PIC X(66)   VALUE SPACES.

01  NO-ACTION-MSG.
    05  FILLER              PIC X       VALUE SPACE.
    05  FILLER              PIC X(50)   VALUE
        '*** ACTION NOT FOUND ON RECORD IN ISEQL01. FORMAT '.
    05  FILLER              PIC X(49)   VALUE
        'SHOULD BE (COLUMN 1): ACTION=XXXXXXXX   RECEIVED:'.
    05  FILLER              PIC X(33)   VALUE SPACES.

01  NO-RESOURCE-MSG.
    05  FILLER              PIC X       VALUE SPACE.
    05  FILLER              PIC X(49)   VALUE
        '*** RESOURCE RECORD NOT FOUND IN ISEQL01 FOR THE '.
    05  FILLER              PIC X(52)   VALUE
        'GIVEN ACTION RECORD, OR RESOURCE NAME DOES NOT BEGIN'.
    05  FILLER              PIC X(13)   VALUE
        ' IN COLUMN 1.'.
    05  FILLER              PIC X(18)   VALUE SPACES.

01  INVALID-ACTION-MSG.
    05  FILLER              PIC X       VALUE SPACE.
    05  FILLER              PIC X(49)   VALUE
        '*** INVALID ACTION RECEIVED IN ISEQL01. MUST BE: '.
    05  FILLER              PIC X(53)   VALUE
        'CLOSE FILE, DISABLE FILE, DISABLE TRAN, ENABLE TRAN, '.
    05  FILLER              PIC X(30)   VALUE
        'DISABLE PROG, OR ENABLE PROG. '.

01  SWITCHES-SW.
    05  TEST-SW                 PIC X       VALUE 'N'.
    05  COMMAND-EOF-SW          PIC X       VALUE 'N'.
    05  GROUP-EOF-SW            PIC X       VALUE 'N'.
    05  NODATA-SW               PIC X       VALUE 'N'.
    05  SCOPE-FOUND-SW          PIC X       VALUE 'N'.
    05  CPSM-CONNECT-SW         PIC X       VALUE 'N'.
        88  CPSM-CONNECT-SUCCEEDED          VALUE 'Y'.
    05  NEW-ACTION-FOUND-SW     PIC X       VALUE 'N'.
        88  NEW-ACTION-FOUND                VALUE 'Y'.
    05  NEW-CONTEXT-FOUND-SW    PIC X       VALUE 'N'.
        88  NEW-CONTEXT-FOUND               VALUE 'Y'.
```

```
    05  PARM-REQUIRED-SW        PIC X       VALUE 'Y'.
    05  INVALID-SCOPE-SW        PIC X       VALUE 'N'.

01  ACCUMULATORS-AA.
    05  COMMAND-RECORDS-READ-AA PIC 9(4)    VALUE ZERO.
    05  GROUP-RECORDS-READ-AA   PIC 9(5)    VALUE ZERO.
    05  RES-OR-GRPS-PROC-AA     PIC 9(5)    VALUE ZERO.
    05  RESOURCES-IN-GROUP-AA   PIC 9(5)    VALUE ZERO.
    05  POS-AA                  PIC 9(2)    VALUE ZERO.

01  MESSAGES-SA.
    05  SCOPE-UNAVAIL-MSG-SA.
        10  FILLER              PIC X(46)   VALUE
            '*** SCOPE UNAVAILABLE - UNABLE TO PROCESS THE '.
        10  FILLER              PIC X(27)   VALUE
            'GIVEN RESOURCE FOR OBJECT('.
    05  CHECK-LOG-MSG-SA.
        10  FILLER              PIC X(49)   VALUE
            '!!! CHECK THE CICS LOGS, CMAS LOG, OR SYSTEM LOG '.
        10  FILLER              PIC X(39)   VALUE
            'FOR ERRORS RELATED TO THIS RESOURCE !!!'.
    05  RSRC-NOT-FOUND-MSG-SA   PIC X(49)   VALUE
        '*** UNABLE TO FIND THE GIVEN RESOURCE FOR OBJECT('.
    05  PROCESS-FAILED-MSG-SA   PIC X(51)   VALUE
        '*** ERROR PROCESSING THE GIVEN RESOURCE FOR OBJECT('.
    05  PROCESS-SUCCESS-MSG-SA  PIC X(30)   VALUE
        'ACTION SUCCESSFUL FOR OBJECT ('.
    05  NO-RESOURCES-PROC-SA    PIC X(31)   VALUE
        '*** NO RESOURCES WERE PROCESSED'.
    05  NO-RES-FOR-GRP-MSG-SA.
        10  FILLER              PIC X(48)   VALUE
            '*** NO RESOURCES WERE FOUND FOR THE GROUP GIVEN '.
        10  FILLER              PIC X(10)   VALUE
            'IN ISEQL02'.
    05  CONN-SUCCESS-MSG-SA     PIC X(44)   VALUE
        'CONNECT COMPLETED SUCCESSFULLY....          '.
    05  CONN-FAILED-MSG-SA.
        10  FILLER              PIC X(43)   VALUE
            '*** UNABLE TO CONNECT TO CPSM FOR THE GIVEN'.
        10  FILLER              PIC X(8)    VALUE
            ' CONTEXT'.
    05  DISCONN-SUCCESS-MSG-SA  PIC X(44)   VALUE
        'DISCONNECT COMPLETED SUCCESSFULLY....       '.
    05  DISCONN-FAILED-MSG-SA.
        10  FILLER              PIC X(43)   VALUE
            '*** UNABLE TO DISCONNECT FROM CPSM FOR THE '.
        10  FILLER              PIC X(13)   VALUE
            'GIVEN CONTEXT'.
    05  GROUP-NOT-FOUND-MSG-SA.
        10  FILLER              PIC X(39)   VALUE
```

```
            '*** THE GROUP DOES NOT EXIST IN ISEQLØ2'.
        1Ø  FILLER              PIC X(39)   VALUE
            ', IS OUT OF SEQUENCE IN ISEQLØ1, OR WAS'.
        1Ø  FILLER              PIC X(42)   VALUE
            ' ALREADY PROCESSED FOR A DIFFERENT REGION.'.
    Ø5  USE-GENERIC-SCOPE-MSG-SA.
        1Ø  FILLER              PIC X(41)   VALUE
            '    USE A MORE GENERIC SCOPE OR RUN THIS '.
        1Ø  FILLER              PIC X(36)   VALUE
            'PROGRAM A SECOND TIME FOR THE OTHER '.
        1Ø  FILLER              PIC X(21)   VALUE
            'REGION, IF APPLICABLE'.

Ø1  SAVE-AREAS-SA.
    Ø5  CPSM-CONTEXT-SA         PIC X(8)    VALUE SPACES.
    Ø5  CPSM-SCOPE-SA           PIC X(8)    VALUE SPACES.
    Ø5  CPSM-OBJNAME-SA         PIC X(8)    VALUE SPACES.
    Ø5  CPSM-ACTION-SA          PIC X(12)   VALUE SPACES.
    Ø5  CPSM-CRITERIA-SA        PIC X(17)   VALUE SPACES.
    Ø5  CPSM-PARM-SA            PIC X(11)   VALUE SPACES.
    Ø5  CPSM-CRITLEN-SA         PIC S9(8)   COMP VALUE ZERO.
    Ø5  CPSM-PARMLEN-SA         PIC S9(8)   COMP VALUE ZERO.
    Ø5  CPSM-THREAD-SA          PIC S9(8)   COMP VALUE ZERO.
    Ø5  CPSM-RESP-SA            PIC S9(8)   COMP VALUE ZERO.
    Ø5  CPSM-REAS-SA            PIC S9(8)   COMP VALUE ZERO.
    Ø5  CPSM-RESULT-SA          PIC S9(8)   COMP VALUE ZERO.
    Ø5  TEMP-RSRC-SA            PIC X(8)    VALUE SPACES.
    Ø5  CRS-ACTION-SA           PIC X(12)   VALUE SPACES.
    Ø5  SYSTEM-CCYYMMDD-DATE-SA PIC 9(Ø8)   VALUE ZERO.
    Ø5  FILLER REDEFINES SYSTEM-CCYYMMDD-DATE-SA.
        1Ø  SYSTEM-CCYY-SA      PIC 9(Ø4).
        1Ø  SYSTEM-MM-SA        PIC 9(Ø2).
        1Ø  SYSTEM-DD-SA        PIC 9(Ø2).
    Ø5  RETURN-CODE-SA          PIC 9(4)    VALUE ZERO.
    Ø5  EYUDA-NUM-SA            PIC 9(4)    VALUE ZERO.
    Ø5  EYUDA-CHAR-SA REDEFINES EYUDA-NUM-SA
                                PIC X(4).
    Ø5  RESP-FIRST-NUM-SA       PIC 9(4)    VALUE 1Ø24.
    Ø5  RESP-NUM-ENTRIES-SA     PIC 9(4)    VALUE 18.
    Ø5  REAS-FIRST-NUM-SA       PIC 9(4)    VALUE 128Ø.
    Ø5  REAS-NUM-ENTRIES-SA     PIC 9(4)    VALUE 85.

Ø1  SUBSCRIPTS-SS.
    Ø5  SUB-SS                  PIC 9(5)    VALUE ZERO.

*****************************************************************
*  WHEN CHANGING THE NUMBER OF OCCURRENCES OF THE RESPONSE TABLE*
*  OR REASON TABLE, CHANGE THE NUM-ENTRIES-SA VARIABLE ABOVE.   *
*  WHEN CHANGING THE STARTING ERROR NUMBER, CHANGE THE FIRST-   *
*  NUM-SA VARIABLE.                                             *
```

```
*****************************************************************
 Ø1  RESPONSE-LIST.
     Ø5  FILLER      PIC X(17)   VALUE 'OK           1024'.
     Ø5  FILLER      PIC X(17)   VALUE 'SCHEDULED    1025'.
     Ø5  FILLER      PIC X(17)   VALUE 'NOTFOUND     1026'.
     Ø5  FILLER      PIC X(17)   VALUE 'NODATA       1027'.
     Ø5  FILLER      PIC X(17)   VALUE 'INVALIDPARM  1028'.
     Ø5  FILLER      PIC X(17)   VALUE 'FAILED       1029'.
     Ø5  FILLER      PIC X(17)   VALUE '             1030'.
     Ø5  FILLER      PIC X(17)   VALUE 'NOTPERMIT    1031'.
     Ø5  FILLER      PIC X(17)   VALUE '             1032'.
     Ø5  FILLER      PIC X(17)   VALUE 'SERVERGONE   1033'.
     Ø5  FILLER      PIC X(17)   VALUE 'NOTAVAILABLE 1034'.
     Ø5  FILLER      PIC X(17)   VALUE 'VERSIONINVL  1035'.
     Ø5  FILLER      PIC X(17)   VALUE 'INVALIDCMD   1036'.
     Ø5  FILLER      PIC X(17)   VALUE 'WARNING      1037'.
     Ø5  FILLER      PIC X(17)   VALUE 'TABLEERROR   1038'.
     Ø5  FILLER      PIC X(17)   VALUE 'INCOMPATIBLE 1039'.
     Ø5  FILLER      PIC X(17)   VALUE 'INUSE        1040'.
     Ø5  FILLER      PIC X(17)   VALUE 'INVALIDATA   1041'.
 Ø1  RESPONSE-RED REDEFINES RESPONSE-LIST.
     Ø5  RESPONSE-TB OCCURS 18 TIMES.
         1Ø  RESPONSE-TEXT-TB  PIC X(12).
         1Ø  FILLER            PIC X(1).
         1Ø  RESPONSE-CODE-TB  PIC 9(4).

 Ø1  REASON-LIST.
     Ø5  FILLER      PIC X(17)   VALUE 'THREAD       1280'.
     Ø5  FILLER      PIC X(17)   VALUE 'OBJECT       1281'.
     Ø5  FILLER      PIC X(17)   VALUE 'CONTEXT      1282'.
     Ø5  FILLER      PIC X(17)   VALUE 'RESULT       1283'.
     Ø5  FILLER      PIC X(17)   VALUE 'COUNT        1284'.
     Ø5  FILLER      PIC X(17)   VALUE 'LENGTH       1285'.
     Ø5  FILLER      PIC X(17)   VALUE 'FILTER       1286'.
     Ø5  FILLER      PIC X(17)   VALUE 'NOTFILTER    1287'.
     Ø5  FILLER      PIC X(17)   VALUE 'FORWARD      1288'.
     Ø5  FILLER      PIC X(17)   VALUE 'BACKWARD     1289'.
     Ø5  FILLER      PIC X(17)   VALUE 'POSITION     1290'.
     Ø5  FILLER      PIC X(17)   VALUE 'DELAY        1291'.
     Ø5  FILLER      PIC X(17)   VALUE 'NOTIFICATION 1292'.
     Ø5  FILLER      PIC X(17)   VALUE 'SIGNONPARM   1293'.
     Ø5  FILLER      PIC X(17)   VALUE 'SCOPE        1294'.
     Ø5  FILLER      PIC X(17)   VALUE 'RESOURCE     1295'.
     Ø5  FILLER      PIC X(17)   VALUE 'FROM         1296'.
     Ø5  FILLER      PIC X(17)   VALUE 'TO           1297'.
     Ø5  FILLER      PIC X(17)   VALUE 'INTO         1298'.
     Ø5  FILLER      PIC X(17)   VALUE 'CRITERIA     1299'.
     Ø5  FILLER      PIC X(17)   VALUE 'BY           1300'.
     Ø5  FILLER      PIC X(17)   VALUE 'ACTION       1301'.
     Ø5  FILLER      PIC X(17)   VALUE 'ECB          1302'.
```

```
05  FILLER      PIC X(17)   VALUE 'SENTINEL     1303'.
05  FILLER      PIC X(17)   VALUE 'FEEDBACK     1304'.
05  FILLER      PIC X(17)   VALUE 'VENT         1305'.
05  FILLER      PIC X(17)   VALUE 'TOKEN        1306'.
05  FILLER      PIC X(17)   VALUE 'MODIFY       1307'.
05  FILLER      PIC X(17)   VALUE 'VIEW         1308'.
05  FILLER      PIC X(17)   VALUE 'FIELDS       1309'.
05  FILLER      PIC X(17)   VALUE 'ATTRIBUTE    1310'.
05  FILLER      PIC X(17)   VALUE 'FROMCV       1311'.
05  FILLER      PIC X(17)   VALUE 'TOCHAR       1312'.
05  FILLER      PIC X(17)   VALUE 'FROMCHAR     1313'.
05  FILLER      PIC X(17)   VALUE 'TOCV         1314'.
05  FILLER      PIC X(17)   VALUE 'PARM         1315'.
05  FILLER      PIC X(17)   VALUE 'PARMLEN      1316'.
05  FILLER      PIC X(17)   VALUE 'SUMOPT       1317'.
05  FILLER      PIC X(17)   VALUE 'TYPE         1318'.
05  FILLER      PIC X(17)   VALUE 'DATALENGTH   1319'.
05  FILLER      PIC X(17)   VALUE 'SOLRESOURCE  1320'.
05  FILLER      PIC X(17)   VALUE 'SOCRESOURCE  1321'.
05  FILLER      PIC X(17)   VALUE 'SOERESOURCE  1322'.
05  FILLER      PIC X(17)   VALUE 'MAINTPOINT   1323'.
05  FILLER      PIC X(17)   VALUE 'SYSNOTACT    1324'.
05  FILLER      PIC X(17)   VALUE 'SYSLVLBAD    1325'.
05  FILLER      PIC X(17)   VALUE 'SYSNOTLCL    1326'.
05  FILLER      PIC X(17)   VALUE 'CICSRELBAD   1327'.
05  FILLER      PIC X(17)   VALUE 'ARMNOTREG    1328'.
05  FILLER      PIC X(17)   VALUE 'ARMNOTACT    1329'.
05  FILLER      PIC X(17)   VALUE 'ARMPOLCHK    1330'.
05  FILLER      PIC X(17)   VALUE 'ABENDED      1331'.
05  FILLER      PIC X(17)   VALUE 'CPSMSYSTEM   1332'.
05  FILLER      PIC X(17)   VALUE 'CPSMVERSION  1333'.
05  FILLER      PIC X(17)   VALUE 'CPSMAPI      1334'.
05  FILLER      PIC X(17)   VALUE 'NOTSUPPORTED 1335'.
05  FILLER      PIC X(17)   VALUE 'NOTVSNCONN   1336'.
05  FILLER      PIC X(17)   VALUE 'INVALIDATTR  1337'.
05  FILLER      PIC X(17)   VALUE 'APITASKERR   1338'.
05  FILLER      PIC X(17)   VALUE 'CPSMSERVER   1339'.
05  FILLER      PIC X(17)   VALUE 'APITASK      1340'.
05  FILLER      PIC X(17)   VALUE 'PLEXMGR      1341'.
05  FILLER      PIC X(17)   VALUE 'REQTIMEOUT   1342'.
05  FILLER      PIC X(17)   VALUE '             1343'.
05  FILLER      PIC X(17)   VALUE 'AREATOOSMALL 1344'.
05  FILLER      PIC X(17)   VALUE 'USRID        1345'.
05  FILLER      PIC X(17)   VALUE '             1346'.
05  FILLER      PIC X(17)   VALUE '             1347'.
05  FILLER      PIC X(17)   VALUE 'VERSION      1348'.
05  FILLER      PIC X(17)   VALUE '             1349'.
05  FILLER      PIC X(17)   VALUE '             1350'.
05  FILLER      PIC X(17)   VALUE '             1351'.
05  FILLER      PIC X(17)   VALUE 'FILTERMATCH  1352'.
```

```
    05  FILLER      PIC X(17)   VALUE 'INVALIDOBJ   1353'.
    05  FILLER      PIC X(17)   VALUE 'INVALIDVER   1354'.
    05  FILLER      PIC X(17)   VALUE 'TASKDATAKEY  1355'.
    05  FILLER      PIC X(17)   VALUE 'INVALIDVERB  1356'.
    05  FILLER      PIC X(17)   VALUE 'NOSTORAGE    1357'.
    05  FILLER      PIC X(17)   VALUE 'NOSERVICE    1358'.
    05  FILLER      PIC X(17)   VALUE 'EXCEPTION    1359'.
    05  FILLER      PIC X(17)   VALUE 'INVALIDEVT   1360'.
    05  FILLER      PIC X(17)   VALUE 'DATAERROR    1361'.
    05  FILLER      PIC X(17)   VALUE 'CMAS         1362'.
    05  FILLER      PIC X(17)   VALUE 'FIRST        1363'.
    05  FILLER      PIC X(17)   VALUE 'NEXT         1364'.
01  REASON-RED REDEFINES REASON-LIST.
    05  REASON-TB OCCURS 85 TIMES.
        10  REASON-TEXT-TB    PIC X(12).
        10  FILLER            PIC X(1).
        10  REASON-CODE-TB    PIC 9(4).

PROCEDURE DIVISION.

000-MAIN.

    OPEN INPUT  COMMAND-FILE-IN
                GROUP-FILE-IN
         OUTPUT  PRINT-FILE-OUT.

    PERFORM 010-CALC-DATE.
    PERFORM 020-PRINT-PAGE-HEADINGS.

    PERFORM 800-READ-COMMAND-FILE.
    IF CRS-CONTEXT-LIT = 'TEST'
        MOVE 'Y' TO TEST-SW
        PERFORM 800-READ-COMMAND-FILE
    END-IF.

    PERFORM 100-PROCESS-COMMANDS
        UNTIL COMMAND-EOF-SW = 'Y' OR
              RETURN-CODE-SA > 4.

    IF RES-OR-GRPS-PROC-AA = ZERO AND
       RETURN-CODE-SA < 5
        MOVE NO-RESOURCES-PROC-SA  TO EL1-ERROR-MSG
        MOVE EL1-ERROR-LINE1       TO PRINT-LINE-OUT
        WRITE PRINT-LINE-OUT AFTER 2
        MOVE 2100                  TO RETURN-CODE-SA
    END-IF.

    PERFORM 030-FINISHING-WORK.
    CLOSE       COMMAND-FILE-IN
                GROUP-FILE-IN
```

```
                PRINT-FILE-OUT.

    MOVE RETURN-CODE-SA TO RETURN-CODE.

    STOP RUN.


Ø1Ø-CALC-DATE.
    CALL 'IGZEDT4' USING BY REFERENCE SYSTEM-CCYYMMDD-DATE-SA.
    MOVE SYSTEM-MM-SA     TO H1-MM.
    MOVE SYSTEM-DD-SA     TO H1-DD.
    MOVE SYSTEM-CCYY-SA   TO H1-CCYY.

Ø2Ø-PRINT-PAGE-HEADINGS.
    MOVE   HEAD-LINE1 TO PRINT-LINE-OUT.
    WRITE  PRINT-LINE-OUT AFTER PAGE.
    MOVE   HEAD-LINE2 TO PRINT-LINE-OUT.
    WRITE  PRINT-LINE-OUT AFTER 2.

Ø3Ø-FINISHING-WORK.
    MOVE 'END OF REPORT' TO EL1-ERROR-MSG.
    MOVE EL1-ERROR-LINE1 TO PRINT-LINE-OUT.
    WRITE PRINT-LINE-OUT AFTER 3.

1ØØ-PROCESS-COMMANDS.
    MOVE 'N' TO CPSM-CONNECT-SW.
    PERFORM 11Ø-FIND-CONTEXT-SCOPE.

    IF RETURN-CODE-SA < 5
        PERFORM 12Ø-EDIT-CONTEXT-SCOPE
    END-IF.

    IF RETURN-CODE-SA < 5
        PERFORM 13Ø-CONNECT-TO-CPSM
    END-IF.

    IF RETURN-CODE-SA < 5
*       READ THE FIRST ACTION RECORD FOR THE CONTEXT
        PERFORM 8ØØ-READ-COMMAND-FILE

        IF COMMAND-EOF-SW = 'Y'
            MOVE NO-ACTION-MSG        TO PRINT-LINE-OUT
            WRITE PRINT-LINE-OUT AFTER 2
            MOVE '    END OF FILE'    TO EL1-ERROR-MSG
            MOVE EL1-ERROR-LINE1      TO PRINT-LINE-OUT
            WRITE PRINT-LINE-OUT AFTER 1
            MOVE 15Ø3 TO RETURN-CODE-SA
        END-IF

        PERFORM 2ØØ-PROCESS-ACTIONS
```

```
            UNTIL NEW-CONTEXT-FOUND  OR
                  RETURN-CODE-SA > 4 OR
                  COMMAND-EOF-SW EQUAL 'Y'
    END-IF.

    IF CPSM-CONNECT-SUCCEEDED
        PERFORM 140-DISCONNECT-FROM-CPSM
    END-IF.

 110-FIND-CONTEXT-SCOPE.
    IF CRS-CONTEXT-LIT NOT EQUAL 'CONTEXT='
        MOVE NO-CONTEXT-MSG TO PRINT-LINE-OUT
        WRITE PRINT-LINE-OUT AFTER 2
        MOVE CRS-COMMAND-RECORD    TO EL1-ERROR-MSG
        MOVE EL1-ERROR-LINE1       TO PRINT-LINE-OUT
        WRITE PRINT-LINE-OUT AFTER 1
        MOVE 1501 TO RETURN-CODE-SA
    ELSE
*       THE SCOPE COMES AFTER THE CONTEXT. ASSUME IT STARTS
*       IN BYTE 9 OR LATER OF CRS-CONTEXT-SCOPE.
        MOVE 'N' TO SCOPE-FOUND-SW
        PERFORM VARYING POS-AA FROM 9 BY 1
            UNTIL SCOPE-FOUND-SW = 'Y' OR
                  POS-AA > 66
            IF CRS-CONTEXT-SCOPE (POS-AA:6) = 'SCOPE='
                MOVE 'Y' TO SCOPE-FOUND-SW
            END-IF
        END-PERFORM
        IF SCOPE-FOUND-SW = 'N'
            MOVE NO-SCOPE-MSG TO PRINT-LINE-OUT
            WRITE PRINT-LINE-OUT AFTER 2
            MOVE CRS-COMMAND-RECORD  TO EL1-ERROR-MSG
            MOVE EL1-ERROR-LINE1     TO PRINT-LINE-OUT
            WRITE PRINT-LINE-OUT AFTER 1
            MOVE 1502 TO RETURN-CODE-SA
        ELSE
*           RESET THE NEW-CONTEXT SWITCH FOR CONTROL BREAK
*           PROCESSING
            MOVE 'N'                     TO NEW-CONTEXT-FOUND-SW
            MOVE CRS-CONTEXT-SCOPE (1:8) TO CPSM-CONTEXT-SA
            ADD 5                        TO POS-AA
            MOVE CRS-CONTEXT-SCOPE (POS-AA:8) TO CPSM-SCOPE-SA
        END-IF
    END-IF.
```

Editor's note: this article will be concluded in next month's issue.

Randy Schulting
Technical Services Analyst
CUNA Mutual Group (USA)

Allen Systems Group has announced Version 2.7 of its ASG-CATS CICS-centralized control tool, which now supports CICS up to and including CICS TS 1.3. It's designed to decrease system downtime and application backlog and increase programmer productivity.

The tool lets system programmers propagate table changes to an unlimited number of regions across local and remote sites. It automates the CICS table change process, standardizing and checking the syntax of all table changes submitted by programmers. These changes are then verified and approved by the system administrators. Once approved, ASG-CATS will then create, assemble, and propagate the changes to all the necessary tables.

In the case of a CICS system failure, managers can access changes and conduct either a full or partial back-out of table changes made during a particular period.

For further information contact:
Allen Systems Group, 750 11th Street South, Naples, FL 34102, USA.
Tel: (941) 435 2200.
URL: http://www.asg.com.

* * *

Computer Associates has announced Unicenter TNG for OS/390, which includes agents for CICS, OS/390 Unix Services, CA-IDMS, CA-Datacom, DB2, MQSeries, Lotus Notes/Domino, SAP R3, PeopleSoft, BaaN, Oracle, and Novell NetWare for OS/390.

A key component, says the vendor, is the Distributed State Machine (DSM), described as a management process that centralizes the discovery of, and interactivity with, all objects and resources within an OS/390 management domain.

DSM gathers real-time information from software agents monitoring managed objects, providing policy-driven management, agent control and interaction, Web-enabled administration, and alleged seamless integration across platforms.

For further information contact:
Computer Associates, 909 E Las Colinas Blvd, Irving, TX 75039, USA.
Tel: (435) 342 5224.
Computer Associates, Ditton Park, Riding Court Road, Datchet, Berks, SL3 9LL, UK.
Tel: (01753) 577733.
URL: http://www.cai.com.

* * *

NEON Systems is shipping Version 4.5 of its Shadow Direct and Shadow OS/390 Web Server products. Enhancements include new support for application servers such as Ardent DataStage, BEA Tuxedo, and WebLogic, Forte, and SilverStream Application Server. Other new functions include a Shadow Web Interface for Internet access to the Shadow ISPF facility and expanded support for tools such as PowerBuilder and Visual Basic.

Enhanced CICS support allows incorporation of terminal-attached transactions without modifications to the transaction and there is also expanded support for VSAM and ADABAS data.

For further information contact:
NEON Systems, 14100 Southwest Freeway, #500 Sugarland, TX 77478, USA.
Tel: (281) 491 4200.
URL: http://www.neonsys.com.