208 CICS update

March 2003

In this issue

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
PO Box 350100
Westminster, CO 80035-0100
USA
Telephone: 303 410 9344

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 December 1999 issue, are available separately to subscribers for £16.00 ($24.00) each including postage.

CICS Update on-line

Code from *CICS Update*, and complete issues in Acrobat PDF format, can be downloaded from our Web site at http://www.xephon.com/cics; you will need to supply a word from the printed issue.

Editor

Trevor Eddolls

Contributions

When Xephon is given copyright, articles published in *CICS Update* are paid for at the rate of £170 ($260) per 1000 words and £100 ($160) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of £50 ($80) per 100 lines. In addition, there is a flat fee of £30 ($50) per article. To find out more about contributing an article, without any obligation, please download a copy of our *Notes for Contributors* from www.xephon.com/nfc.

CICS automatic restart

There have been many articles published about automatic restart in the last few years – way back in 1995 I wrote an article for *CICS Update* that was published in the September issue (118). I have reviewed each article that has appeared since, but have not really found any major differences or improvements to this solution. Looking at recent articles, it appears that some sites are not quite clear when a restart of CICS should (must) be done to ensure data integrity.

Actually, as far as I understand, a CICS system has to be checked immediately after it has terminated to determine whether CICS has written a warm keypoint and terminated normally. If this warm keypoint is missing then the CICS is 'in flight', and inconsistent datasets (VSAM, DL/I, or DB2) may exist. CICS will automatically correct this (if everything is defined correctly) if it is started again with an emergency restart (naturally ESDS datasets are an exception). This should be done in any case to ensure that the databank and VSAM datasets are all recovered and that all necessary transaction backouts are processed. If this is not done, then any processing to the datasets by batch may change data that needs to be backed out – in other words you can destroy or change the data for the CICS backout, so that CICS cannot recover properly! Even saving the databases and VSAM would be saving inconsistent data. In my opinion, it is not possible to wait until a later time for an emergency CICS restart, if any changes to the data occur between the termination and restart of the CICS system.

After saying all this, I would like to emphasize that the automatic restart system that we use covers the problems mentioned above because the checks are done after CICS termination, and processing continues only after the warm keypoint has been validated – otherwise CICS will restart and the emergency restart will clarify all open issues (backouts). This is completely automated at our site, and, if CICS crashes, it is restarted so fast that we are rarely even notified of a problem. One problem that can occur with this system is CICS crashing each time it is started. You can have a yoyo effect – it starts, crashes, the job does not find a warm keypoint so restarts CICS, it crashes, etc. This rarely happens and the operators know that they can just cancel the restart job to stop the yoyo effect.

With CICS TS the warm keypoint has completely changed. Previously (CICS Versions 3 and 4) the warm keypoint was written to the global catalog dataset. In CICS Version 5 (TS) the warm keypoint is now handled by the recovery manager domain. An entirely new key is written to the global catalog dataset.

Example of a warm keypoint that was taken:

```
KEY OF RECORD -
ØØØØØØ11C4C6C8D9D4C4D44ØC4C6C8D9D4C4D46DC1D5C3C8D6D94Ø4Ø
 ØØØØØØ  ØØØØØØ11 C4C6C8D9 D4C4D44Ø C4C6C8D9 *....DFHRMDM DFHR*
 ØØØØ1Ø  D4C4D46D C1D5C3C8 D6D94Ø4Ø ØØ1ØC4C5 *MDM_ANCHOR  ..DE*
 ØØØØ2Ø  C9C2D4C9 E54BC9E5 F4C1F5F3 C1F1ØØØ3 *IBMIV.IV4A53A1..*
 ØØØØ3Ø  ØØØ4ØØØØ ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ *................*
 ØØØØ4Ø  ØØØØB4FC A753ØDA5 A1Ø4ØØØØ ØØØØØØØØ *................*
 ØØØØ5Ø  ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ          *............    *
```

Example of an emergency restart that is required:

```
 KEY OF RECORD -
ØØØØØØ11C4C6C8D9D4C4D44ØC4C6C8D9D4C4D46DC1D5C3C8D6D94Ø4Ø
 ØØØØØØ  ØØØØØØ11 C4C6C8D9 D4C4D44Ø C4C6C8D9 *....DFHRMDM DFHR*
 ØØØØ1Ø  D4C4D46D C1D5C3C8 D6D94Ø4Ø ØØ1ØC4C5 *MDM_ANCHOR  ..DE*
 ØØØØ2Ø  C9C2D4C9 E54BC9E5 F4C1F5F3 C1F1ØØØ4 *IBMIV.IV4A53A1..*
 ØØØØ3Ø  ØØØ4ØØØØ ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ *................*
 ØØØØ4Ø  ØØØØB5Ø1 DF4EF266 FFØ4ØØØØ ØØØØØØØØ *.....+2.........*
 ØØØØ5Ø  ØØØØØØØØ ØØØØØØØØ ØØØØØØØØ          *............    *
 .
```

Here is the modified program that we now use for CICS TS. The jobs and other information can be extracted from Issues 118 and 120 (a minor correction to 118). The best way to search for the articles on the Xephon Web site (to get fewest hits) is to search for the author’s name (Mongan).

```
       IDENTIFICATION DIVISION.
       PROGRAM-ID.
           SYSPGØ3Ø.
```

```
 AUTHOR.
     K. MONGAN.
 DATE-WRITTEN.
     FEB   1994.
*REMARKS.

*    Read the DFHGCD to check if an EMERGENCY RESTART
*    is necessary – if yes then give a RETURN CODE 16

 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
     DECIMAL-POINT IS COMMA.
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*       D A T A  -  D I V I S I O N                     *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 INPUT-OUTPUT SECTION.
 FILE-CONTROL.
     SELECT  DFHGCD          ASSIGN  TO  SYSØ1Ø-DFHGCD
                             ORGANIZATION INDEXED
                             ACCESS MODE  RANDOM
                             RECORD KEY   SCHLUESSEL IN DFHGCDS
                             FILE STATUS  CHK.
 DATA DIVISION.
 FILE SECTION.

 FD  DFHGCD
     RECORD CONTAINS 28 TO 4Ø89 CHARACTERS
     DATA   RECORD  DFHGCDS.

 Ø1  DFHGCDS.
     Ø2  SCHLUESSEL          PIC X(28).
     Ø2  FILLER              PIC X(11).
     Ø2  APPLID              PIC X(8).
     Ø2  EMERKZ              PIC X.
 Ø1  FILLER                  PIC X(28).
 Ø1  FILLER                  PIC X(4Ø89).

     EJECT
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
*                                                       *
*       WORKING-STORAGE                                 *
*                                                       *
* * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 WORKING-STORAGE SECTION.

 Ø1  FZEILE.
     Ø2  FTEXT               PIC  X(3Ø)
                             VALUE "PROG-CANCEL FILE-STATUS = ".
```

```
           Ø2  CHK                 PIC  XX    VALUE SPACE.

       Ø1  FZEILE2.
           Ø2  FTEXT2              PIC X(8Ø)  VALUE SPACE.

       Ø1  WS-SCHL.
           Ø2  FILLER              PIC X(3)   VALUE LOW-VALUE.
           Ø2  FILLER              PIC X(1)   VALUE X"11".
           Ø2  FILLER              PIC X(8)   VALUE "DFHRMDM ".
           Ø2  FILLER              PIC X(14)  VALUE "DFHRMDM_ANCHOR".
           Ø2  FILLER              PIC X(2)   VALUE SPACE.


      * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      *       LINKAGE SECTION                                 *
      * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
       LINKAGE SECTION.

           EJECT
      * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
      *       PROCEDURE DIVISION                              *
      * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
       PROCEDURE DIVISION.

       ANFANG.
           OPEN INPUT                        DFHGCD .
           IF   CHK                 NOT =    "ØØ"
                MOVE "OPEN ERROR" TO FTEXT2
                GO TO EMER-ROUTINE.
           MOVE WS-SCHL                 TO   KEYS IN DFHGCDS.
           READ DFHGCD.
           IF   CHK                 NOT =    "ØØ"
                MOVE "READ ERROR" TO FTEXT2
                GO TO EMER-ROUTINE.

           IF   EMERKZ  IN DFHGCDS  NOT =    X"Ø3"
                GO TO EMER-ROUTINE.
      *ENDE-ROUTINE.
           MOVE ZERO                    TO   RETURN-CODE.
           CLOSE DFHGCD .
           DISPLAY "CICS ENDED    —>  SYSPGØ3Ø " APPLID UPON CONSOLE
           GOBACK.

       EMER-ROUTINE.
           MOVE "16"    TO RETURN-CODE
           MOVE "EMERGENCY RESTART NEEDED FS = " TO FTEXT.

       ABBRUCH-ROUTINE.
           DISPLAY FZEILE                                  UPON CONSOLE
           DISPLAY FZEILE2                                 UPON CONSOLE
```

```
          DISPLAY "          "                         UPON CONSOLE
          DISPLAY "CICS CANCELLED —>  SYSPGØ3Ø " APPLID UPON CONSOLE
          GOBACK.
```

William Kim Mongan
Systems Programmer (Germany)

Repeat lines in BMS sources

When you have a map with several lines repeated, where the only thing that changes is the line number, and perhaps the field's name or line label, editing the BMS can be a boring task. Normally, you create the source lines that represent the fields for a given line in the final screen, and then you repeat those lines the desired number of times.

The boring part is to go through the repeated source lines and adjust the line number in the POS(xx,yy) parameter, and perhaps the field's name or line labels, where they have one.

To avoid this, I created an edit macro that repeats lines and at the same time adjusts the POS parameter and the line labels. The format of the macro is as follows:

```
REPEAT first_line last_line times increment
```

First_line and *last_line* are the absolute line numbers of the block to repeat. *Times* is the number of repetitions to perform. If omitted, it defaults to 1. *Increment* represents the line jump, and is meaningful only when the logical block being repeated spans more than one screen line, or if you want to leave a blank line in the screen. If omitted, increment is 1.

The POS(xx,yy) parameter must be specified in such a way that it can hold the greatest number that will be generated. For example, an initial POS=(3,7) will not be repeatable ten times, because the last value would be POS(12,7). This means I would have to stretch the line by 1 byte, which could cause problems if the line was already full or contained literals. To avoid this, it is your responsibility to specify POS=(03,7) in such situations.

As for the labels or field names, I only change the names if they end up with a number, otherwise I will not touch them. If they end with a number, the same rule applies – they must already have enough digits to hold the largest number that will be generated.

A small example of labels and how they will be handled:

- NAME – this label will not be changed because it ends with a letter.
- NAME1 – this label will be incremented (NAME2, NAME3, etc) provided it does not go beyond NAME9.
- NAME01 – this label will be repeatable up to 98 times.

The increment value for labels is always 1, even if specified differently. Increment is intended for screen line numbers, when the logical block being repeated spans more than one screen line. However, the fields within that block are 1, in the sense that they refer to the same logical unit of information, so it would make no sense to increment them in the same way lines must be.

A FEW EXAMPLES OF HOW THINGS WORK

Consider the following piece of source, which contains three fields, all of them in the same screen line. To triplicate these fields, issue the command:

```
Command ===> repeat 176 18Ø 3                            Scroll ===> CSR
 ØØØ176 NAMEØ1   DFHMDF POS=(Ø5,Ø2),LENGTH=3Ø,ATTRB=(PROT),
*
 ØØØ177                 HILIGHT=OFF,COLOR=YELLOW
 ØØØ178 DESCØ1   DFHMDF POS=(Ø5,33),LENGTH=2Ø,ATTRB=(PROT),
*
 ØØØ179                 HILIGHT=OFF,COLOR=BLUE
 ØØØ18Ø          DFHMDF POS=(Ø5,54),LENGTH=Ø1
 ØØØ181 *
```

and the result will be:

```
 ØØØ176 NAMEØ1   DFHMDF POS=(Ø5,Ø2),LENGTH=3Ø,ATTRB=(PROT),
*
```

```
 ØØØ177                 HILIGHT=OFF,COLOR=YELLOW
 ØØØ178 DESCØ1   DFHMDF POS=(Ø5,33),LENGTH=2Ø,ATTRB=(PROT),
*
 ØØØ179                 HILIGHT=OFF,COLOR=BLUE
 ØØØ18Ø          DFHMDF POS=(Ø5,54),LENGTH=Ø1
 ØØØ181 NAMEØ2   DFHMDF POS=(Ø6,Ø2),LENGTH=3Ø,ATTRB=(PROT),
*
 ØØØ182                 HILIGHT=OFF,COLOR=YELLOW
 ØØØ183 DESCØ2   DFHMDF POS=(Ø6,33),LENGTH=2Ø,ATTRB=(PROT),
*
 ØØØ184                 HILIGHT=OFF,COLOR=BLUE
 ØØØ185          DFHMDF POS=(Ø6,54),LENGTH=Ø1
 ØØØ186 NAMEØ3   DFHMDF POS=(Ø7,Ø2),LENGTH=3Ø,ATTRB=(PROT),
*
 ØØØ187                 HILIGHT=OFF,COLOR=YELLOW
 ØØØ188 DESCØ3   DFHMDF POS=(Ø7,33),LENGTH=2Ø,ATTRB=(PROT),
*
 ØØØ189                 HILIGHT=OFF,COLOR=BLUE
 ØØØ19Ø          DFHMDF POS=(Ø7,54),LENGTH=Ø1
 ØØØ191 *
```

Now consider a group of three fields that span two screen lines (3 and 4). To ‘quadruplicate’ them, you must now specify the increment factor of 2, in order to make source line 3 be repeated to line 5, and source line 4 go to line 6, and so on. As explained before, labels are always increased by 1.

```
Command ===> repeat 176 18Ø 4 2                        Scroll ===> CSR
 ØØØ176 CODEØ1   DFHMDF POS=(Ø3,Ø2),LENGTH=Ø8,ATTRB=(PROT)
 ØØØ177 NAMEØ1   DFHMDF POS=(Ø3,15),LENGTH=45,ATTRB=(PROT),
*
 ØØØ178                 HILIGHT=OFF,COLOR=YELLOW
 ØØØ179 DESCØ1   DFHMDF POS=(Ø4,11),LENGTH=6Ø,ATTRB=(PROT),
*
 ØØØ18Ø                 HILIGHT=OFF,COLOR=BLUE
```

In this case, the result will be:

```
 ØØØ176 CODEØ1   DFHMDF POS=(Ø3,Ø2),LENGTH=Ø8,ATTRB=(PROT)
 ØØØ177 NAMEØ1   DFHMDF POS=(Ø3,15),LENGTH=45,ATTRB=(PROT),
*
 ØØØ178                 HILIGHT=OFF,COLOR=YELLOW
 ØØØ179 DESCØ1   DFHMDF POS=(Ø4,11),LENGTH=6Ø,ATTRB=(PROT),
*
 ØØØ18Ø                 HILIGHT=OFF,COLOR=BLUE
 ØØØ181 CODEØ2   DFHMDF POS=(Ø5,Ø2),LENGTH=Ø8,ATTRB=(PROT)
 ØØØ182 NAMEØ2   DFHMDF POS=(Ø5,15),LENGTH=45,ATTRB=(PROT),
*
 ØØØ183                 HILIGHT=OFF,COLOR=YELLOW
 ØØØ184 DESCØ2   DFHMDF POS=(Ø6,11),LENGTH=6Ø,ATTRB=(PROT),
*
 ØØØ185                HILIGHT=OFF,COLOR=BLUE
 ØØØ186 CODEØ3   DFHMDF POS=(Ø7,Ø2),LENGTH=Ø8,ATTRB=(PROT)
 ØØØ187 NAMEØ3   DFHMDF POS=(Ø7,15),LENGTH=45,ATTRB=(PROT),
*
 ØØØ188                HILIGHT=OFF,COLOR=YELLOW
 ØØØ189 DESCØ3   DFHMDF POS=(Ø8,11),LENGTH=6Ø,ATTRB=(PROT),
*
 ØØØ19Ø                HILIGHT=OFF,COLOR=BLUE
 ØØØ191 CODEØ4   DFHMDF POS=(Ø9,Ø2),LENGTH=Ø8,ATTRB=(PROT)
 ØØØ192 NAMEØ4   DFHMDF POS=(Ø9,15),LENGTH=45,ATTRB=(PROT),
*
 ØØØ193                HILIGHT=OFF,COLOR=YELLOW
 ØØØ194 DESCØ4   DFHMDF POS=(1Ø,11),LENGTH=6Ø,ATTRB=(PROT),
*
 ØØØ195                HILIGHT=OFF,COLOR=BLUE
```

REPEAT SOURCE CODE

```
/*== REXX  ISPF editor macro =====================================*/
/*                                                                */
/*   REPEAT firstline lastline <times> <increment>                */
/*                                                                */
/*================================================================*/

 address ISPEXEC
'ISREDIT MACRO (ARG)'
'ISREDIT (LASTLINE) = linenum .zlast'
 upper arg
 line1 = word(arg,1)
 line2 = word(arg,2)
 times = word(arg,3)
 incre = word(arg,4)
 if line1 = "" then signal helpe
 if times = "" then times = 1
 if incre = "" then incre = 1

 if datatype(line1,"W") & datatype(line2,"W") & ,
    datatype(times,"W") & datatype(incre,"W")
    then nop
 else do
    zedlmsg="Arguments must be numeric"
    address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
    exit 99
 end
 if line2 < line1 then do
    zedlmsg="Last line cannot be less than first line"
    address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
```

```
   exit 99
end
if line2 > lastline then do
   zedlmsg="Last line greater than number of lines of file"
   address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   exit 99
end
if times < 1 then do
   zedlmsg="Minimum number of times is 1"
   address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   exit 99
end
if incre < 1 then do
   zedlmsg="Minimum increment is 1"
   address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   exit 99
end
if incre * times > 999 then do
   zedlmsg="Maximum times multiplied by incre must be less than 999"
   address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
   exit 99
end

do k = line1 to line2
  'isredit (linetext) = line' k
   lin.k = linetext
   po = pos('POS=(',lin.k)
   if po > Ø then do
      pospos.k = po + 5
      su = substr(lin.k,pospos.k,5)
      parse var su s "," .
      poslen.k = length(s)
      posval.k = substr(lin.k,pospos.k,poslen.k)
      dt = datatype(posval.k)
      if dt <> "NUM" then do
         zedlmsg=" Position parameter not numeric at lineno" k
         address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
         exit 99
      end
      test = posval.k + times * incre
      if test > 9  & poslen.k < 2 |,
         test > 99 & poslen.k < 3 then do
         zedlmsg="Not enough digits for POS parameter at lineno" k
         address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
         exit 99
      end
      changeline.k = k
   end

   if left(lin.k,1) <> " " then do
```

```
        label.k = word(lin.k,1)
        lablen.k = Ø
        do z = 1 to 3
           dt = datatype(right(label.k,z))
           if dt = "NUM" then do
              lablen.k = z
           end
        end
        if lablen.k = Ø then iterate k
        changelabel.k = k
        labpos.k = length(label.k) - lablen.k + 1
        labval.k = right(label.k,lablen.k)
        test = labval.k + times
        if test > 9  & lablen.k < 2 |,
           test > 99 & lablen.k < 3 then do
           zedlmsg="Not enough digits for LABEL at lineno" k
           address ISPEXEC 'SETMSG MSG(ISRZØØ1)'
           exit 99
        end
     end
  end

  current_line = line2
  do j = 1 to times
     do k = line1 to line2
        if k = changeline.k then do
           posval.k = posval.k + incre
           posval.k = right(posval.k,poslen.k,"Ø")
           lin.k = overlay(posval.k,lin.k,pospos.k)
        end
        if k = changelabel.k then do
           labval.k = labval.k + 1
           labval.k = right(labval.k,lablen.k,"Ø")
           lin.k = overlay(labval.k,lin.k,labpos.k)
        end
        'ISREDIT LINE_AFTER' current_line '= "'lin.k'"'
         current_line = current_line + 1
     end
  end

exit

/*=================================================================*/
helpe:
say "REPEAT firstline lastline <times> <increment>                    "
say
say " repeats lines from firstline to lastline times. Times is the    "
say " number of times you want to replicate the original block.       "
say " The default is 1.                                               "
say
```

```
say "This macro is aimed at BMS data sources. The line position number"
say "and, optionally, the labels found within the block specified are "
say "automatically incremented (default 1), provided that the original"
say "number contains enough digits to accommodate the largest         "
say "incremented number.                                              "
say "In the case of labels, they are only incremented if they end up  "
say "with a number, otherwise they will not be changed.               "
say "Their increment value is always 1.                               "
exit
```

Systems Programmer
(Portugal) © Xephon 2003

CICSPlex SM dynamic workload management – usage

In this article we will look at practical usage of the features provided by CICS TS. Unlike the academic approach where you can start from scratch, in real life you have many other things to contend with. The applications you want to workload manage probably already exist, they probably have affinities that you don't know about, and you've probably not got the source anymore. If you do have the source, the application programmers probably have 'better things to do' than removing affinities. Furthermore I've yet to meet a CIO who will allow you to shut everything down for several weeks whilst you do this. No, they must continue to execute during the transition to dynamic routing. So, how do you go about it?

ESTABLISHING A SIMPLE SET-UP

CICSPlex SM can support many sophisticated configurations. Here we are looking at a simple set-up (and common pitfalls) on the CICSPlex SM side of things. I'm assuming the connections and sessions are appropriately set up, and the dynamic routing attributes are appropriately set. The procedure is:

- Make sure that EYU9XLOP is specified as DTRPGM in all routers for this type of workload (check via CICSRGN).
- Make sure EYU9XLOP is specified as DSRTPGM in all routers and targets for this type of workload (check via CICSRGN).
- If you have specified DTRTRAN in the SIT, then dynamic routing will be invoked for any transaction that is not found in a router. If EYU9XLOP is invoked and there isn't a workload registered for this router, then CICSPLex SM will assume that the CMAS has yet to come up and associate a workload with this region. It will therefore wait. You will get a message 'Waiting for workload' in the console log. So:
  – Ensure that workload specifications are associated with all routers (you can check with WLMSCOPE and WLMATOR).
  – Ensure that all target regions are identified as targets in the appropriate workloads (check via MAP on WLMSPEC and WLMAAOR).
  – For DSRTPGM workloads the routers and targets must be in the same workload (check via WLMSCOPE).
- Ensure that routing is active at startup (Defined in CSYSDEF records).
- Check WLM is active (via MAS view).
- You can also check your configuration via the WLMATOR, WLMAAOR, and WLMAWORK views.

UNDERSTAND YOUR SYSTEM CONFIGURATION

The simplest part of workload management is setting up the definitions and installing them. If only life were so simple. You need to understand your present (and future) system configuration. You need to understand the operational procedures that operators employ today and what they must move to at each step of the way. For each step of the move, you must establish failure procedures and back-up procedures. You must design your new configuration and how you're going to get there.

If possible, define new regions for your dynamic routing regions. It'll be easier to debug if something goes wrong. If you have 'spaghetti' combined TOR/AOR regions, consider separating TOR and AOR regions, with static routing as a first step (yes, that will increase overhead, but it's simpler if you can).

UNDERSTAND YOUR APPLICATIONS

Understanding your applications is absolutely essential to success. Use the affinities and interdependency utilities to identify affinities. Try to move one application at a time. Take 'baby steps' to get some success. If you have resource available to remove affinities, everything else being equal, you should try to remove the longest lifetime affinities first (balanced least frequently). Also, eliminate affinities that cause broadcasts (reduced exposure to CMAS failure and better performance – see below).

CONSIDER THE 'NO TARGETS AVAILABLE' SCENARIO

What are you going to do if CICSPlex SM finds that all target regions are not available?

A common solution to this scenario is to default routing to the router. This is achieved by providing no remote attributes on the resource definitions for the transaction etc.

If you have a situation where the router is not a target, then you can make the transaction definition reference a **different** program in the router. This program will inform the end user why they can't use the regular one at the moment.

CONSIDER LOCKING REQUIREMENTS OF THE APPLICATION

Consider the following dynamic DPL scenario. A program uses DB2. On its first route it routes to region 1 and references a DB2 table causing locks to be obtained. On a subsequent invocation it is routed to a different region where it tries to access the same data. Since locks are still owned by the program in region 1, the program deadlocks. A similar situation would occur with recoverable TD queues.

A way out of this is to define affinity or create/destroy affinity dynamically. On the first invocation, affinity would be created to region 1. Subsequent routings would also go to region 1 (where the locks are held) since affinity is active. Destroy the affinity when the end of the Unit Of Work completes.

THINKING ABOUT PERFORMANCE

Some affinities are worse than others. On affinity creation, depending on the relation and lifetime, affinities can require a CICSplex-wide lock to be obtained (ie affinity elements need broadcasting across the CMAS network). If any CMAS in the network is down, the affinity element cannot be created and the transaction cannot be executed. If the affinity element already exists, then it can execute in the absence of the CMAS.

STEPS TO EXPLOITATION

The recommended steps to exploitation are as follows:

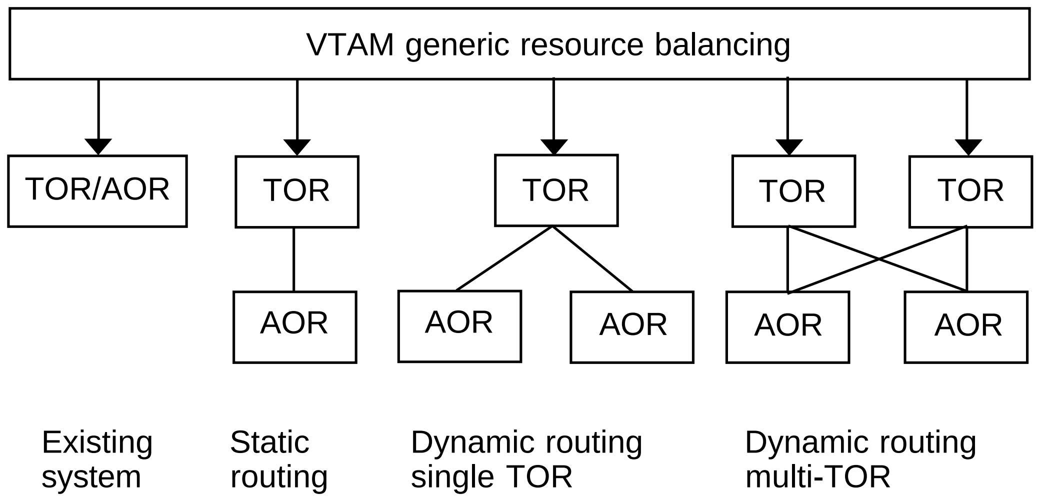
VTAM generic resource balancing
TOR/AOR
TOR
TOR
TOR
TOR
AOR
AOR
AOR
AOR
AOR
Existing system
Static routing
Dynamic routing single TOR
Dynamic routing multi-TOR

Figure 1: Resource balancing

- Understand you system configuration and plan your steps to exploitation.
- Understand your application affinities.
- Split combined TOR/AORs to static routing.
- Adopt VTAM generic resource for increased TOR availability.
- Clone your AORs for availability and adopt dynamic workload management.
- Clone your TORs for increased availability.

Resource balancing is illustrated in Figure 1.

WHAT DYNAMIC ROUTING CAN'T COPE WITH

The ability to route work dynamically in a CICSplex provides high availability and reliability; however, it is not a silver bullet. Dynamic routing is in essence an external function that acts with no understanding of the internals of a given piece of work. All it has to work on are response times and maybe the odd abend to signal a problem. Most programming standards (quite rightly) tell the programmer not to abend, but to provide a meaningful message back to the user and a tidy termination ("Database is unavailable – Please retry later"). To workload management, this is a successful termination; and a fast one at that. Such 'soft failures' are undetectable, although some failures in DB2 are reflected back to CICS, allowing workload management to detect and take appropriate action.

Another class of work is so called 'killer transactions'. If you have a transaction that is so badly written that it manages to take the CICS region down, you have trouble. If you are dynamically routing it to all your AORs, then you get into the sequence of 'route to 1; 1 abends, route to 2, 2 abends, etc'. You get the picture? Your whole set of AORs dies. If you have some of these applications, get them fixed. In the interim you can use workload separation to route to a different set of AORs for these transactions and switch storage isolation on.

Perhaps information is available in the transaction's TIOA, COMMAREA, or TCTUA. If it is, consider customizing the CICSPlex SM routing exit to control execution (we will see how to do this in the next article).

OPERATIONAL CONTROL

CICSPlex SM provides the ability to display:

- Active workloads – workloads active within the CICSplex.
- Active routing regions – identifies routing regions for a workload.
- Active target regions – identifies target regions for a workload. Target regions can be quiesced/activated.
- Active workload definitions – active workload separation definitions. These can be discarded (but check affinities first).
- Active affinities – shows affinities active for a workload. These can be discarded – be aware of consequences!
- Transaction and trangrps associated with workload.

NEXT ARTICLE

In the last article we will look at exit customization.

Dr Paul Johnson
CICS Transaction Server Systems Management Planning/Development
IBM (UK)

Journal buffer flushing and CICS shutdown assist

INTRODUCTION

CICS Transaction Server journal management has a subtle difference in its buffer flushing implementation when compared with that of CICS/ESA 4.1.0 and below. This article discusses the background to this in relation to the CICS-supplied shutdown assist mechanism.

BACKGROUND

CICS Log Manager support was introduced in CICS TS 1.1 as a replacement for the journal control program DFHJCP (and associated modules), which had provided logging support in CICS/ESA 4.1.0 and earlier releases. DFHJCP handled logging requests to BSAM sequential datasets defined to tape or (more commonly) as dual disk extents.

The Log Manager was provided as a new object-oriented Domain within CICS Transaction Server, which utilizes the MVS System Logger to write and retrieve log data for both the system log and user journals. The system log is composed of two underlying logstreams – DFHLOG and DFHSHUNT. User journals map on to general logstreams; these are for use as audit trails, forward recovery journals, logging of TIOA data, etc.

When a CICS/ESA 4.1.0 system was shut down the CICS System Termination Program DFHSTP would invoke the Journal Control Shutdown program DFHJCSDJ to terminate the journalling environment. Part of this work would involve closing the journal for each BSAM journal dataset. This close would implicitly flush any data remaining in the CICS journal buffers. That is, all unforced journal records in the buffers were ‘hardened’ to BSAM by I/O activity during the close. DFHJCSDJ was invoked by DFHSTP for both controlled and uncontrolled CICS shutdowns – that is, initiated by either CEMT PERFORM SHUT or CEMT PERFORM SHUT IMMEDIATE commands or their EXEC CICS equivalents.

It should be noted that an MVS cancel of the CICS region would not have driven CICS/ESA 4.1.0 system termination processing, nor invoked DFHJCSDJ.

In CICS Transaction Server, DFHSTP no longer invokes journalling services during shutdown processing. The CICS Log Manager services are invoked by CICS Domain Management as part of system shutdown. In the case of a normal shutdown (CEMT PERFORM SHUT), CICS Log Manager services are invoked to quiesce the CICS journalling environment. Outstanding unflushed data in the journal buffers is flushed to the MVS Logger, and the logstreams are closed. MVS Logger services are invoked (via IXGCONN DISCONNECT calls) to disconnect the CICS system from the logstreams.

In the case of an immediate shutdown, however, (CEMT PERFORM SHUTDOWN IMMEDIATE), CICS Domain Management does not perform the housekeeping activity described above. The logstreams are not closed, and IXGCONN calls are not issued to disconnect CICS from the logstreams (the disconnections happen during MVS End Of Task processing for the CICS Address Space). Immediate shutdowns are intended to provide a quick exit from the CICS environment; they do not perform the same controlled quiesce function as a controlled CICS shutdown, such as the invocation of MVS System Logger services to disconnect from a potentially larger number of logstreams.

Despite not flushing the journal buffer contents during an immediate CICS shutdown, CICS system data integrity is not threatened. Any system log data within the journal buffers for the system logstream DFHLOG (or DFHSHUNT) will have been physically written to the logstream by CICS at the point when it needed to be hardened, ie prior to any changes to recoverable resources. This means that unflushed log data within the system log buffers is not necessary to reconstruct inflight changes to any recoverable resources that have been made, when the system is subsequently emergency restarted and inflight work backed out. This is the same situation as when an emergency restart is performed following a power failure, machine check, or MVS cancel of the region. These circumstances do not provide the opportunity for flushing system log data, yet, for the reason given above, CICS data integrity is not affected by this.

Similarly, CICS file control forward recovery journalling is not jeopardized by any of its associated journal buffers not being flushed during an immediate shutdown. This is because CICS ensures that forward recovery log data (after-images of changes made to CICS-managed VSAM files) is physically written to its associated logstream when the task that makes the changes syncpoints and so commits them. Any forward recovery data within the journal buffers at the time of an immediate shutdown reflects work performed by a task or tasks that are still inflight at the time, and so have yet to commit their changes via syncpoint. This means that these changes will be backed out by CICS as part of the subsequent emergency restart, and hence data integrity will be maintained.

The difference between CICS Transaction Server and CICS/ ESA 4.1.0 is that any general logstream data remaining in the journal buffer, pertaining to user journals, cannot be guaranteed to have been written to the underlying logstream when CICS Transaction Server is terminated with an immediate shutdown. At worst this means that the last log block of data is not available on the logstream for the journal after CICS has been terminated. Again, it should be noted that this same situation can have occurred in previous releases of the product, should a CICS system be lost because of external circumstances, such as a machine failure or MVS cancel.

As stated in the *CICS Recovery and Restart Guide*, immediate shutdowns are not recommended. As a general rule when terminating CICS, a normal (controlled) shutdown is the recommended approach; an immediate shutdown should be used only if there is a special reason for doing so.

CICS SHUTDOWN ASSIST

CICS provides a shutdown assist mechanism to help actively manage the termination of a CICS system. The System Initialization Parameter SDTRAN may be used to specify the name of a shutdown assistance transaction to be attached during system termination. The CICS-supplied transid CESD is the default shutdown assistance transaction, and DFHCESD the default Assembler program to be run. DFHCESD attempts to purge and back out long-running tasks using increasingly stronger techniques. It ensures that as many tasks as possible complete their activity and commit or backout their changes cleanly, thus enabling CICS to be shutdown in a controlled manner if possible. If this is not possible for some reason, preventing a controlled CICS shutdown from completing, the shutdown assistance program can issue an EXEC CICS PERFORM SHUTDOWN IMMEDIATE command to ensure that the system terminates.

Note that users may replace this default program and transaction with their own; samples DFH£CESD and SDA1, and DFH0CESD and SDA2 are provided for this purpose. DFH£CESD is written in PL/I and DFH0CESD in COBOL.

Given that shutdown assistance can result in an immediate shutdown, there is the possibility that unhardened user journal data may be lost as a result, for the reasons explained above. In order to avoid this, users may wish to use a PLTSD Phase 1 program to issue EXEC CICS SET JOURNAL FLUSH commands for their user journals. This will ensure that the CICS Log Manager journal buffers are written to their corresponding logstreams. Note that PLTSD Phase 1 programs are invoked before an immediate shutdown is initiated by the shutdown assist transaction. Alternatively, users may wish to modify the sample shutdown assistance programs to issue SET JOURNAL FLUSH commands for their user journals, prior to issuing an immediate shutdown. This has the advantage of hardening any buffered journal data written by tasks that completed during Phase 1 of shutdown.

An alternative solution is to review the writing of user journal data by applications and consider the use of the WAIT option on the CICS journal commands. WAIT will ensure that log data is written to the underlying journal logstream synchronously with the request, rather than buffered by CICS and written asynchronously at a later stage. Another option is to issue an EXEC CICS WAIT JOURNALNAME command directly within the application program, to drive I/O for the journal buffer (although, being performed after the data was written to the buffer, this option cannot guarantee that an immediate shutdown or system failure will not occur between then and when the data was originally written to the buffer). If program changes are not viable, an end-of-task TRUE may be considered. This could issue EXEC CICS SET JOURNAL FLUSH commands for the appropriate journals utilized by the finishing task, to ensure that their journal data is hardened if required.

Analysis of user journal usage can reveal those journals that are used explicitly by applications (via EXEC CICS WRITE JOURNAL commands) and those implicitly written to by CICS (eg autojournalling for files or terminal control I/O logging). Given this information, approaches such as modifying the shutdown assist samples, coding WAIT commands, or using end-of-task TRUE programs to issue flushes may be considered if required.

SUMMARY AND CONTACT INFORMATION

It should be emphasized that immediate shutdown processing does not threaten CICS system data integrity, for either system resources logged to the CICS system log or file control after-images written to forward recovery journals.

I hope that this article has been useful in explaining the options available to help ensure that any user journal data is written to its underlying logstream in the event of an immediate CICS shutdown. For further details, please refer to the chapter headed *Shutdown and restart recovery* in the CICS Transaction Server *Recovery and Restart Guide*.

Andy Wright (andy_wright@uk.ibm.com)
CICS Change Team
IBM (UK)

CICSPlex SM API – REXX EXECs (run-time interface) – part 2

This month we publish the remaining REXX EXECs to use the CICSPlex SM API run-time interface.

CPSMTSQI

CPSMTSQI displays information about CICS TSQs based on the context, scope, and criteria specified. The CICSPlex SM TSQNAME resource table has a number of attributes that could be used as criteria. The CPSMTSQI example shows that there are 64 bytes of information available for each TSQ.

```
/****************************** REXX ********************************/
/*  MODULE NAME : CPSMTSQI                                          */
/*  MODULE TYPE : REXX Executable                                   */
/*  DESCRIPTION : CICSPlex SM  -  TSQNAME Resource Information      */
/*                Displays information about TSQNAME resources      */
/*                based on the parameters provided.                 */
/*  INVOCATION  : CPSMTSQI w_context w_scope w_criteria             */
/*                w_context = name of a CMAS or CICSplex            */
/*                w_scope   = name of CICSplex, CICS system group   */
/*                            or CICS system within w_context       */
/*                w_criteria= the criteria to be used to filter     */
/*                            the TSQNAME objects                   */
/*  RETURN      : retc      = return code                           */
/********************************************************************/
Trace
rexx_name = 'CPSMTSQI'
retc = Ø
w_context = ''
w_scope = ''
w_criteria = ''
msg = 'Invoked on 'DATE()' at 'TIME()'.'
say rexx_name msg
Parse Upper Arg w_context w_scope w_criteria
if w_context = '' | w_scope = '' | w_criteria = ''
  then
    do
      msg = 'Context, scope, and criteria MUST be specified.'
      say rexx_name msg
      retc = 8
      Signal RETURN_CONTROL
    end
msg = 'Invoked with parameters:-'
say rexx_name msg
msg = 'Context  : 'w_context
say rexx_name msg
msg = 'Scope    : 'w_scope
say rexx_name msg
msg = 'Criteria : 'w_criteria
say rexx_name msg
/********************************************************************/
/*  Call external subroutine CPSMINIT to initialize the CICSPlex SM  */
/*  REXX/API environment. If the return code from CPSMINIT isn't     */
/*  zero then terminate with the return code.                        */
/*********************************************************************/
Call CPSMINIT
  if result <> Ø
    then
      do
        retc = result
        Signal RETURN_CONTROL
      end
/*********************************************************************/
/*  Call external subroutine CPSMCONN to establish a connection to   */
/*  CICSPlex SM using the context and scope specified. If the        */
/*  return code from CPSMCONN isn't zero then terminate the          */
/*  CICSPlex SM REXX/API environment and terminate this REXX with    */
/*  the return code. If the return code is zero use the CICSPlex SM  */
/*  token "w_thread" for all subsequent commands.                    */
/*********************************************************************/
Call CPSMCONN w_context, w_scope
Parse Var result w_retc w_thread .
  if w_retc <> Ø
    then
      do
        retc = w_retc
        Call CPSMTERM
        Signal RETURN_CONTROL
      end
/*********************************************************************/
/*  GET the TSQNAME objects based on the context, scope, and         */
/*  criteria specified at invocation. If the CICSPlex SM response    */
/*  is not OK and not NODATA then issue diagnostic messages and set  */
/*  return code before disconnecting and terminating. A response of  */
/*  NODATA is not an error as this response indicates that no        */
/*  TSQNAME objects matched the criteria specified within the        */
/*  context and scope. Issue an approriate message, disconnect       */
/*  and terminate. If the response is OK then process the TSQNAME    */
/*  records.                                                         */
/*********************************************************************/
w_criteria = w_criteria||"."
w_criterialen = 'LENGTH'(w_criteria)
Address CPSM 'GET OBJECT(TSQNAME)',
             'CRITERIA(w_criteria)',
             'LENGTH('w_criterialen')',
             'COUNT(w_count)',
             'RESULT(w_result)',
             'THREAD(w_thread)',
             'RESPONSE(w_response)',
             'REASON(w_reason)'
if w_response <> EYURESP(OK)
  then
```

```
    do
      if w_response = EYURESP(NODATA)
        then
          do
            msg = 'No TSQs match the criteria specified'
            say rexx_name msg
          end
        else
          do
            msg = 'Failure GETting OBJECT(TSQNAME):'
            say rexx_name msg
            retc = 16
            msg = 'Response = 'EYURESP(w_response),
                  'Reason = 'EYUREAS(w_reason)
            say rexx_name msg
          end
      Signal CPSM_DISCONNECT
    end
msg = 'GET retrieved 'w_count' OBJECT(TSQNAME) records.'
say rexx_name msg
/*********************************************************************/
/*  Obtain information about the TSQNAME object, ie the length       */
/*  of the object records. If the response code from CICSPlex SM     */
/*  is not OK then issue diagnostic messages and set the return      */
/*  code before disconnecting and terminating. If the response code  */
/*  is OK then issue a message with the number of bytes per record   */
/*  for the object.                                                  */
/*********************************************************************/
Address CPSM 'QUERY OBJECT(TSQNAME)',
             'THREAD(w_thread)',
             'RESULT(w_result)',
             'DATALENGTH(w_into_objectlen)',
             'RESPONSE(w_response)',
             'REASON(w_reason)'
if w_response <> EYURESP(OK)
  then
    do
      msg = 'Failure QUERYing OBJECT(TSQNAME):'
      say rexx_name msg
      retc = 16
      msg = 'Response = 'EYURESP(w_response),
            'Reason = 'EYUREAS(w_reason)
      say rexx_name msg
      Signal CPSM_DISCONNECT
    end
msg = 'Each OBJECT(TSQNAME) record is 'w_into_objectlen' bytes.'
say rexx_name msg
/*********************************************************************/
/*  Loop through the TSQNAME results and translate the output into   */
/*  displayable characters. Issue a message for each TSQNAME. If     */
/*  CICSPlex SM commands receive a response other than OK then      */
/*  issue diagnostic messages and set the return code before        */
/*  disconnecting and terminating.                                  */
/********************************************************************/
do iii = 1 to w_count
  Address CPSM 'FETCH INTO(w_into_object)',
               'LENGTH(w_into_objectlen)',
               'POSITION('iii')',
               'RESULT(w_result)',
               'THREAD(w_thread)',
               'RESPONSE(w_response)',
               'REASON(w_reason)'
  if w_response <> EYURESP(OK)
    then
      do
        msg = 'Failure FETCHing record:'
        say rexx_name msg
        retc = 16
        msg = 'Response = 'EYURESP(w_response),
              'Reason = 'EYUREAS(w_reason)
        say rexx_name msg
        Signal CPSM_DISCONNECT
      end
  Address CPSM 'TPARSE OBJECT(TSQNAME)',
               'PREFIX(tsq)',
               'STATUS(w_response)',
               'VAR(w_into_object.1)',
               'THREAD(w_thread)'
  if w_response <> 'OK'
    then
      do
        msg = 'Failure parsing record. Status='w_response
        say rexx_name msg
        retc = 16
        Signal CPSM_DISCONNECT
      end
  /******************************************************************/
  /*  Format the display information for the TSQNAME results.       */
  /*  cicsname TSQ(                ) TRAN(    ) LOC(    )           */
  /*           INT(        )                                        */
  /******************************************************************/
  msg = '   'tsq_eyu_cicsname' TSQ(                ) TRAN(    )',
        'LOC(    ) INT(ØØØØØØØ)'
  msg = 'OVERLAY'(tsq_name,msg,17)
  msg = 'OVERLAY'(tsq_transid,msg,4Ø)
  msg = 'OVERLAY'(tsq_location,msg,5Ø,4)
  int_pos = 67 - 'LENGTH'(tsq_lastusedint)
  msg = 'OVERLAY'(tsq_lastusedint,msg,int_pos)
  say rexx_name msg
end iii
```

```
CPSM_DISCONNECT:
/*******************************************************************/
/*  Call external subroutine CPSMDISC to disconnect from CICSPlex SM */
/*  using the CICSPlex SM token "w-thread". Regardless of the       */
/*  return code from CPSMDISC termination processing continues      */
/*  normally.                                                        */
/*******************************************************************/
Call CPSMDISC w_thread
/*******************************************************************/
/*  Call external subroutine CPSMTERM to terminate the CICSPlex SM   */
/*  REXX/API environment. Regardless of the return code from        */
/*  CPSMTERM termination processing continues normally.             */
/*******************************************************************/
Call CPSMTERM
RETURN_CONTROL:
/*******************************************************************/
/*  Return control with a return code - this is the only exit point  */
/*******************************************************************/
msg = 'Terminated with return code: 'retc
say rexx_name msg
exit (retc)
```

CPSMTSQI could, for example, be used to display all CICS TSQs with a specific or generic name or a total length greater than 10,000 bytes or any combination of attributes.

The following example displays all CICS TSQs in the CICS system group TEST within the CICSplex TPLEX which have not been referenced for more than 24 hours:

```
CPSMTSQI TPLEX TEST LASTUSEDINT>864ØØ

CPSMTSQI Invoked on 18 Nov 2ØØ2 at 1Ø:31:27.
CPSMTSQI Invoked with parameters:-
CPSMTSQI Context  : TPLEX
CPSMTSQI Scope    : TEST
CPSMTSQI Criteria : LASTUSEDINT>864ØØ
CPSMTSQI GET retrieved 3 OBJECT(TSQNAME) records.
CPSMTSQI Each OBJECT(TSQNAME) record is 64 bytes.
CPSMTSQI    CICSTAØ1 TSQ(XPEDXIVP        ) TRAN(DBPA) LOC(AUXI)
INT(Ø234692)
CPSMTSQI    CICSTAØ2 TSQ(XPEDXIVP        ) TRAN(DBPA) LOC(AUXI)
INT(Ø2347Ø1)
CPSMTSQI    CICSTTØ1 TSQ(XPEDXIVP        ) TRAN(DBPA) LOC(AUXI)
INT(Ø234691)
CPSMTSQI Terminated with return code: Ø
```

CPSMTSQD deletes CICS TSQs based on the context, scope, and criteria specified. CPSMTSQD was specifically written for applications which use the EXEC CICS WRITEQ TS command liberally, but never use the EXEC CICS DELETEQ TS command.

```
/******************************* REXX ********************************/
/*  MODULE NAME : CPSMTSQD                                           */
/*  MODULE TYPE : REXX Executable                                    */
/*  DESCRIPTION : CICSPlex SM  -  Delete TSQNAME Resources           */
/*                Deletes CICS Temporary Storage Queues (TSQ)        */
/*                based on the parameters provided.                  */
/*  INVOCATION  : CPSMTSQD w_context w_scope w_criteria              */
/*                w_context = name of a CMAS or CICSplex             */
/*                w_scope   = name of CICSplex, CICS system group    */
/*                            or CICS system within w_context        */
/*                w_criteria= the criteria to be used to filter      */
/*                            the TSQNAME objects                    */
/*  RETURN      : retc      = return code                            */
/*********************************************************************/
Trace
rexx_name = 'CPSMTSQD'
retc = Ø
w_context = ''
w_scope = ''
w_criteria = ''
msg = 'Invoked on 'DATE()' at 'TIME()'.'
say rexx_name msg
Parse Upper Arg w_context w_scope w_criteria
if w_context = '' | w_scope = '' | w_criteria = ''
  then
    do
      msg = 'Context, scope, and criteria MUST be specified.'
      say rexx_name msg
      retc = 8
      Signal RETURN_CONTROL
    end
msg = 'Invoked with parameters:-'
say rexx_name msg
msg = 'Context  : 'w_context
say rexx_name msg
msg = 'Scope    : 'w_scope
say rexx_name msg
msg = 'Criteria : 'w_criteria
say rexx_name msg
/*********************************************************************/
/*  Call external subroutine CPSMINIT to initialize the CICSPlex SM  */
/*  REXX/API environment. If the return code from CPSMINIT isn't     */
/*  zero then terminate with the return code.                          */
/*********************************************************************/
Call CPSMINIT
  if result <> Ø
    then
      do
        retc = result
        Signal RETURN_CONTROL
      end
/*********************************************************************/
/*  Call external subroutine CPSMCONN to establish a connection to   */
/*  CICSPlex SM using the context and scope specified. If the        */
/*  return code from CPSMCONN isn't zero then terminate the          */
/*  CICSPlex SM REXX/API environment and terminate this REXX with    */
/*  the return code. If the return code is zero use the CICSPlex SM  */
/*  token "w_thread" for all subsequent commands.                    */
/*********************************************************************/
Call CPSMCONN w_context, w_scope
Parse Var result w_retc w_thread .
  if w_retc <> Ø
    then
      do
        retc = w_retc
        Call CPSMTERM
        Signal RETURN_CONTROL
      end

/*********************************************************************/
/*  PERFORM the ACTION(DELETE) for the OBJECT(TSQNAME) based         */
/*  on context, scope and criteria specified at invocation. If the   */
/*  CICSPlex SM response is not OK and not NODATA then issue         */
/*  diagnostic messages and set return code before disconnecting     */
/*  and terminating. A response of NODATA indicates that no          */
/*  TSQNAME objects matched the criteria specified within the        */
/*  context and scope. Issue an approriate message, disconnect       */
/*  and terminate. If the response is OK then process the TSQNAME    */
/*  records.                                                         */
/*********************************************************************/
w_criteria = w_criteria||"."
w_criterialen = 'LENGTH'(w_criteria)
Address CPSM 'PERFORM OBJECT(TSQNAME)',
             'ACTION(DELETE)',
             'CRITERIA(w_criteria)',
             'LENGTH('w_criterialen')',
             'COUNT(w_count)',
             'RESULT(w_result)',
             'THREAD(w_thread)',
             'RESPONSE(w_response)',
             'REASON(w_reason)'
if w_response <> EYURESP(OK)
```

```
  then
    do
      if w_response = EYURESP(NODATA)
        then
          do
            msg = 'No TSQs match the criteria specified'
            say rexx_name msg
          end
        else
          do
            msg = 'Failure PERFORMing ACTION(DELETE):'
            say rexx_name msg
            retc = 16
            msg = 'Response = 'EYURESP(w_response),
                  'Reason = 'EYUREAS(w_reason)
            say rexx_name msg
          end
      Signal CPSM_DISCONNECT
    end
msg = 'PERFORMed ACTION(DELETE) for 'w_count' TSQs.'
say rexx_name msg
/*******************************************************************/
/*  Obtain information about the TSQNAME object, ie the length      */
/*  of the object records. If the response code from CICSPlex SM    */
/*  is not OK then issue diagnostic messages and set the return     */
/*  code before disconnecting and terminating. If the response code  */
/*  is OK then issue a message with the number of bytes per record   */
/*  for the object.                                                  */
/*******************************************************************/
Address CPSM 'QUERY OBJECT(TSQNAME)',
             'THREAD(w_thread)',
             'RESULT(w_result)',
             'DATALENGTH(w_into_objectlen)',
             'RESPONSE(w_response)',
             'REASON(w_reason)'
if w_response <> EYURESP(OK)
  then
    do
      msg = 'Failure QUERYing OBJECT(TSQNAME):'
      say rexx_name msg
      retc = 16
      msg = 'Response = 'EYURESP(w_response),
            'Reason = 'EYUREAS(w_reason)
      say rexx_name msg
      Signal CPSM_DISCONNECT
    end
msg = 'Each OBJECT(TSQNAME) record is 'w_into_objectlen' bytes.'
say rexx_name msg
/*******************************************************************/
/*  Loop through the TSQNAME results and translate the output into   */
/*  displayable characters. Issue a message for each TSQNAME. If      */
/*  CICSPlex SM commands receive a response other than OK then        */
/*  issue diagnostic messages and set the return code before          */
/*  disconnecting and terminating.                                    */
/**********************************************************************/
do iii = 1 to w_count
  Address CPSM 'FETCH INTO(w_into_object)',
               'LENGTH(w_into_objectlen)',
               'POSITION('iii')',
               'RESULT(w_result)',
               'THREAD(w_thread)',
               'RESPONSE(w_response)',
               'REASON(w_reason)'
  if w_response <> EYURESP(OK)
    then
      do
        msg = 'Failure FETCHing record:'
        say rexx_name msg
        retc = 16
        msg = 'Response = 'EYURESP(w_response),
              'Reason = 'EYUREAS(w_reason)
        say rexx_name msg
        Signal CPSM_DISCONNECT
      end
  Address CPSM 'TPARSE OBJECT(TSQNAME)',
               'PREFIX(tsq)',
               'STATUS(w_response)',
               'VAR(w_into_object.1)',
               'THREAD(w_thread)'
  if w_response <> 'OK'
    then
      do
        msg = 'Failure parsing record. Status='w_response
        say rexx_name msg
        retc = 16
        Signal CPSM_DISCONNECT
      end
  /********************************************************************/
  /*  Format the display information for the TSQNAME results.         */
  /*  cicsname TSQ(                ) TRAN(    ) LOC(    )             */
  /*           INT(        )                                          */
  /********************************************************************/
  msg = '   'tsq_eyu_cicsname' TSQ(                ) TRAN(    )',
        'LOC(    ) INT(ØØØØØØØ)'
  msg = 'OVERLAY'(tsq_name,msg,17)
  msg = 'OVERLAY'(tsq_transid,msg,4Ø)
  msg = 'OVERLAY'(tsq_location,msg,5Ø,4)
  int_pos = 67 - 'LENGTH'(tsq_lastusedint)
  msg = 'OVERLAY'(tsq_lastusedint,msg,int_pos)
  say rexx_name msg
```

```
end iii
CPSM_DISCONNECT:
/*******************************************************************/
/*  Call external subroutine CPSMDISC to disconnect from CICSPlex SM */
/*  using the CICSPlex SM token "w-thread". Regardless of the       */
/*  return code from CPSMDISC termination processing continues      */
/*  normally.                                                       */
/*******************************************************************/
Call CPSMDISC w_thread
/*******************************************************************/
/*  Call external subroutine CPSMTERM to terminate the CICSPlex SM   */
/*  REXX/API environment. Regardless of the return code from        */
/*  CPSMTERM termination processing continues normally.             */
/*******************************************************************/
Call CPSMTERM
RETURN_CONTROL:
/*******************************************************************/
/*  Return control with a return code - this is the only exit point  */
/*******************************************************************/
msg = 'Terminated with return code: 'retc
say rexx_name msg
exit (retc)
```

The following example deletes all CICS TSQs in the CICS system group TEST within the CICSplex TPLEX that were created by a transaction not beginning with ‘D’ and have not been referenced for more than 24 hours:

```
CPSMTSQD TPLEX TEST TRANSID^=D* AND LASTUSEDINT>864ØØ

CPSMTSQD Invoked on 18 Nov 2ØØ2 at 1Ø:35:51.
CPSMTSQD Invoked with parameters:-
CPSMTSQD Context  : TPLEX
CPSMTSQD Scope    : TEST
CPSMTSQD Criteria : TRANSID^=D* AND LASTUSEDINT>864ØØ
CPSMTSQD No TSQs match the criteria specified
CPSMTSQD Terminated with return code: Ø
```

CPSMNEWC

CPSMNEWC performs a CICS program PHASEIN based on the context, scope, and criteria specified. The CICSPlex SM PROGRAM resource table has a number of attributes that could be used as criteria. The CPSMNEWC example shows that 408 bytes of information are available for each program instance.

```
/****************************** REXX ********************************/
```

```
/*  MODULE NAME : CPSMNEWC                                           */
/*  MODULE TYPE : REXX Executable                                    */
/*  DESCRIPTION = CICSPlex SM  -  Program PHASEIN                    */
/*                Performs a program PHASEIN based on the            */
/*                parameters provided.                               */
/*  INVOCATION  : CPSMNEWC w_context w_scope w_criteria              */
/*                w_context = name of a CMAS or CICSplex             */
/*                w_scope   = name of CICSplex, CICS system group    */
/*                            or CICS system within w_context        */
/*                w_criteria= the criteria to be used to filter      */
/*                            the PROGRAM objects                    */
/*  RETURN      : retc      = return code                            */
/*********************************************************************/
Trace
rexx_name = 'CPSMNEWC'
retc = Ø
w_context = ''
w_scope = ''
w_criteria = ''
msg = 'Invoked on 'DATE()' at 'TIME()'.'
say rexx_name msg
Parse Upper Arg w_context w_scope w_criteria
if w_context = '' | w_scope = '' | w_criteria = ''
  then
    do
      msg = 'Context, scope, and criteria MUST be specified.'
      say rexx_name msg
      retc = 8
      Signal RETURN_CONTROL
    end
msg = 'Invoked with parameters:-'
say rexx_name msg
msg = 'Context  : 'w_context
say rexx_name msg
msg = 'Scope    : 'w_scope
say rexx_name msg
msg = 'Criteria : 'w_criteria
say rexx_name msg
/*********************************************************************/
/*  Call external subroutine CPSMINIT to initialize the CICSPlex SM  */
/*  REXX/API environment. If the return code from CPSMINIT isn't     */
/*  zero then terminate with the return code.                        */
/*********************************************************************/
Call CPSMINIT
  if result <> Ø
    then
      do
        retc = result
        Signal RETURN_CONTROL
      end
```

```
/*******************************************************************/
/*  Call external subroutine CPSMCONN to establish a connection to   */
/*  CICSPlex SM using the context and scope specified. If the        */
/*  return code from CPSMCONN isn't zero then terminate the          */
/*  CICSPlex SM REXX/API environment and terminate this REXX with    */
/*  the return code. If the return code is zero use the CICSPlex SM  */
/*  token "w_thread" for all subsequent commands.                    */
/*******************************************************************/
Call CPSMCONN w_context, w_scope
Parse Var result w_retc w_thread .
  if w_retc <> Ø
    then
      do
        retc = w_retc
        Call CPSMTERM
        Signal RETURN_CONTROL
      end
/*******************************************************************/
/*  PERFORM the ACTION(PHASEIN) for the OBJECT(PROGRAM) based        */
/*  on context, scope and criteria specified at invocation. If the   */
/*  CICSPlex SM response is not OK and not NODATA then issue         */
/*  diagnostic messages and set return code before disconnecting     */
/*  and terminating. A response of NODATA indicates that no          */
/*  PROGRAM objects matched the criteria specified within the        */
/*  context and scope. Issue an appropriate message, disconnect      */
/*  and terminate. If the response is OK then process the PROGRAM    */
/*  records.                                                         */
/*******************************************************************/
w_criteria = w_criteria||"."
w_criterialen = 'LENGTH'(w_criteria)
Address CPSM 'PERFORM OBJECT(PROGRAM)',
             'ACTION(PHASEIN)',
             'CRITERIA(w_criteria)',
             'LENGTH('w_criterialen')',
             'COUNT(w_count)',
             'RESULT(w_result)',
             'THREAD(w_thread)',
             'RESPONSE(w_response)',
             'REASON(w_reason)'
if w_response <> EYURESP(OK)
  then
    do
      if w_response = EYURESP(NODATA)
        then
          do
            msg = 'ACTION(PHASEIN) not required for 'w_criteria
            say rexx_name msg
          end
        else
          do
```

```
            msg = 'Failure PERFORMing ACTION(PHASEIN):'
            say rexx_name msg
            retc = 16
            msg = 'Response = 'EYURESP(w_response),
                  'Reason = 'EYUREAS(w_reason)
            say rexx_name msg
          end
      Signal CPSM_DISCONNECT
    end
msg = 'PERFORMed ACTION(PHASEIN) for 'w_count' programs.'
say rexx_name msg
/*********************************************************************/
/*  Obtain information about the PROGRAM object, ie the length       */
/*  of the object records. If the response code from CICSPlex SM     */
/*  is not OK then issue diagnostic messages and set the return      */
/*  code before disconnecting and terminating. If the response code  */
/*  is OK then issue a message with the number of bytes per record   */
/*  for the object.                                                  */
/*********************************************************************/
Address CPSM 'QUERY OBJECT(PROGRAM)',
             'THREAD(w_thread)',
             'RESULT(w_result)',
             'DATALENGTH(w_into_objectlen)',
             'RESPONSE(w_response)',
             'REASON(w_reason)'
if w_response <> EYURESP(OK)
  then
    do
      msg = 'Failure QUERYing OBJECT(PROGRAM):'
      say rexx_name msg
      retc = 16
      msg = 'Response = 'EYURESP(w_response),
            'Reason = 'EYUREAS(w_reason)
      say rexx_name msg
      Signal CPSM_DISCONNECT
    end
msg = 'Each OBJECT(PROGRAM) record is 'w_into_objectlen' bytes.'
say rexx_name msg
/*********************************************************************/
/*  Loop through the PROGRAM results and translate the output into   */
/*  displayable characters. Issue a message for each PROGRAM. If     */
/*  CICSPlex SM commands receive a response other than OK then       */
/*  issue diagnostic messages and set the return code before         */
/*  disconnecting and terminating.                                   */
/*********************************************************************/
do iii = 1 to w_count
  Address CPSM 'FETCH INTO(w_into_object)',
               'LENGTH(w_into_objectlen)',
               'POSITION('iii')',
               'RESULT(w_result)',
             'THREAD(w_thread)',
             'RESPONSE(w_response)',
             'REASON(w_reason)'
  if w_response <> EYURESP(OK)
    then
      do
        msg = 'Failure FETCHing record:'
        say rexx_name msg
        retc = 16
        msg = 'Response = 'EYURESP(w_response),
              'Reason = 'EYUREAS(w_reason)
        say rexx_name msg
        Signal CPSM_DISCONNECT
      end
  Address CPSM 'TPARSE OBJECT(PROGRAM)',
             'PREFIX(program)',
             'STATUS(w_response)',
             'VAR(w_into_object.1)',
             'THREAD(w_thread)'
  if w_response <> 'OK'
    then
      do
        msg = 'Failure parsing record. Status='w_response
        say rexx_name msg
        retc = 16
        Signal CPSM_DISCONNECT
      end
  /*******************************************************************/
  /*  Format the display information for the PROGRAM results.        */
  /*  cicsname PROGRAM(        ) LANG(            )                  */
  /*           LEN(          ) DFHRPL(  )                            */
  /*******************************************************************/
  msg = '   'program_eyu_cicsname' PROGRAM(        )',
        'LANG(            ) LEN(ØØØØØØØØØØ) DFHRPL(ØØ)'
  msg = ,OVERLAY'(program_program,msg,21)
  msg = 'OVERLAY'(program_language,msg,36)
  len_pos = 64 - 'LENGTH'(program_length)
  msg = 'OVERLAY'(program_length,msg,len_pos)
  rpl_pos = 75 - 'LENGTH'(program_rplid)
  msg = 'OVERLAY'(program_rplid,msg,rpl_pos)
  say rexx_name msg
end iii
CPSM_DISCONNECT:
/*********************************************************************/
/*  Call external subroutine CPSMDISC to disconnect from CICSPlex SM */
/*  using the CICSPlex SM token "w-thread". Regardless of the        */
/*  return code from CPSMDISC termination processing continues       */
/*  normally.                                                        */
/*********************************************************************/
Call CPSMDISC w_thread
```

```
/*********************************************************************/
/*  Call external subroutine CPSMTERM to terminate the CICSPlex SM   */
/*  REXX/API environment. Regardless of the return code from         */
/*  CPSMTERM termination processing continues normally.              */
/*********************************************************************/
Call CPSMTERM
RETURN_CONTROL:
/*********************************************************************/
/*  Return control with a return code - this is the only exit point  */
/*********************************************************************/
msg = 'Terminated with return code: 'retc
say rexx_name msg
exit (retc)
```

The following example performs a PHASEIN for programs in the CICS system group TEST within the CICSplex TPLEX with the specific name TS400:

```
CPSMNEWC TPLEX TEST PROGRAM=TS4ØØ

CPSMNEWC Invoked on 18 Nov 2ØØ2 at 1Ø:39:22.
CPSMNEWC Invoked with parameters:-
CPSMNEWC Context  : TPLEX
CPSMNEWC Scope    : TEST
CPSMNEWC Criteria : PROGRAM=TS4ØØ
CPSMNEWC PERFORMed ACTION(PHASEIN) for 3 programs.
CPSMNEWC Each OBJECT(PROGRAM) record is 4Ø8 bytes.
CPSMNEWC    CICSTAØ1 PROGRAM(TS4ØØ   ) LANG(ASSEMBLER   )
LEN(ØØØØØØ328Ø) DFHRPL(ØØ)
CPSMNEWC    CICSTAØ2 PROGRAM(TS4ØØ   ) LANG(ASSEMBLER   )
LEN(ØØØØØØ328Ø) DFHRPL(ØØ)
CPSMNEWC    CICSTTØ1 PROGRAM(TS4ØØ   ) LANG(ASSEMBLER   )
LEN(ØØØØØØ328Ø) DFHRPL(ØØ)
CPSMNEWC Terminated with return code: Ø
```

CPSMAPGM

CPSMAPGM was written to solve a problem when using:

```
CPSMNEWC TPLEX TPLEX PROGRAM=*
```

which always resulted in TABLEERROR and DATAERROR. CPSMNEWC could not establish which programs had been successfully phased in and which had failed. CPSMAPGM processes the feedback information from CICSPlex SM and displays only the programs for which PHASEIN failed with the CICS function, response, and reason codes. The criterion has been hard-coded and is intended to select application programs only.

```
/****************************** REXX *********************************/
/*  MODULE NAME : CPSMAPGM                                           */
/*  MODULE TYPE : REXX Executable                                    */
/*  DESCRIPTION : CICSPlex SM  -  PHASEIN ALL Application Programs   */
/*                Performs a program PHASEIN for ALL                 */
/*                programs except those excluded in                  */
/*                the criteria specified.                            */
/*  INVOCATION  : CPSMAPGM w_context w_scope                         */
/*                w_context = name of a CMAS or CICSplex             */
/*                w_scope   = name of CICSplex, CICS system group    */
/*                            or CICS system within w_context        */
/*  RETURN      : retc      = return code                            */
/*********************************************************************/
Trace
rexx_name = 'CPSMAPGM'
retc = Ø
w_context = ''
w_scope = ''
feedback_flag = 'OFF'
feedback_count = Ø
msg = 'Invoked on 'DATE()' at 'TIME()'.'
say rexx_name msg
Parse Upper Arg w_context w_scope .
if w_context = '' | w_scope = ''
  then
    do
      msg = 'Context and scope MUST be specified.'
      say rexx_name msg
      retc = 8
      signal RETURN_CONTROL
    end
/*********************************************************************/
/*  Program prefixes excluded from PHASEIN.                          */
/*********************************************************************/
w_criteria = 'NOT ('||,
             'PROGRAM=C*',
             'OR PROGRAM=DBUG*',
             'OR PROGRAM=DFH*',
             'OR PROGRAM=DSN*',
             'OR PROGRAM=EYU*',
             'OR PROGRAM=IBM*',
             'OR PROGRAM=IGZ*',
             'OR HOLDSTATUS=HOLD)'
msg = 'Invoked with parameters:-'
say rexx_name msg
msg = 'Context  : 'w_context
say rexx_name msg
```

```
msg = 'Scope    : 'w_scope
say rexx_name msg
msg = 'Criteria : 'w_criteria
say rexx_name msg
/*********************************************************************/
/*  Call external subroutine CPSMINIT to initialize the CICSPlex SM  */
/*  REXX/API environment. If the return code from CPSMINIT isn't     */
/*  zero then terminate with the return code.                        */
/*********************************************************************/
Call CPSMINIT
  if result <> Ø
    then
      do
        retc = result
        Signal RETURN_CONTROL
      end
/*********************************************************************/
/*  Call external subroutine CPSMCONN to establish a connection to   */
/*  CICSPlex SM using the context and scope specified. If the        */
/*  return code from CPSMCONN isn't zero then terminate the          */
/*  CICSPlex SM REXX/API environment and terminate this REXX with    */
/*  the return code. If the return code is zero use the CICSPlex SM  */
/*  token "w_thread" for all subsequent commands.                    */
/*********************************************************************/
Call CPSMCONN w_context, w_scope
Parse Var result w_retc w_thread .
  if w_retc <> Ø
    then
      do
        retc = w_retc
        Call CPSMTERM
        Signal RETURN_CONTROL
      end
/*********************************************************************/
/*  PERFORM the ACTION(PHASEIN) for the OBJECT(PROGRAM) based        */
/*  on context, scope and criteria specified at invocation.          */
/*  If the CICSPlex SM response TABLEERROR and reason DATAERROR      */
/*  occurs FEEDBACK information will be processed. If the            */
/*  CICSPlex SM response is not OK then issue diagnostic messages    */
/*  and set return code before disconnecting and terminating.        */
/*********************************************************************/
w_criteria = w_criteria||"."
w_criterialen = 'LENGTH'(w_criteria)
Address CPSM 'PERFORM OBJECT(PROGRAM)',
             'ACTION(PHASEIN)',
             'CRITERIA(w_criteria)',
             'LENGTH('w_criterialen')',
             'COUNT(w_count)',
             'RESULT(w_result)',
             'THREAD(w_thread)',
             'RESPONSE(w_response)',
             'REASON(w_reason)'
if w_response <> EYURESP(OK)
  then
    do
      if w_response = EYURESP(TABLEERROR) &,
         w_reason   = EYUREAS(DATAERROR)
        then
          do
            feedback_flag = 'ON'
            feedback_length = 1000000
            feedback_count = 10000
            Address CPSM 'FEEDBACK INTO(feedback)',
                         'LENGTH(feedback_length)',
                         'COUNT(feedback_count)',
                         'FIRST',
                         'RESULT(w_result)',
                         'THREAD(w_thread)',
                         'RESPONSE(w_response)',
                         'REASON(w_reason)'
            if w_response <> EYURESP(OK)
              then
                do
                  msg = 'Failure obtaining FEEDBACK:'
                  say rexx_name msg
                  Signal CPSM_DISCONNECT
                end
          end
        else
          do
            msg = 'Failure PERFORMing ACTION(PHASEIN):'
            say rexx_name msg
            retc = 16
            msg = 'Response = 'EYURESP(w_response),
                  'Reason = 'EYUREAS(w_reason)
            say rexx_name msg
            Signal RETURN_CONTROL
          end
    end
success_count = w_count - feedback_count
msg = 'PERFORMed ACTION(PHASEIN) for 'w_count' programs.'
say rexx_name msg
msg = 'ACTION(PHASEIN)',
      'successful for 'success_count' programs.'
say rexx_name msg
msg = 'ACTION(PHASEIN)',
      'failed for 'feedback_count' programs.'
say rexx_name msg
/**********************************************************************/
/*  Obtain information about the PROGRAM object, ie the length       */
/*  of the object records. If the response code from CICSPlex SM     */
/*  is not OK then issue diagnostic messages and set the return      */
/*  code before disconnecting and terminating. If the response code  */
/*  is OK then issue a message with the number of bytes per record   */
/*  for the object. Processed only if FEEDBACK records are present.  */
/*********************************************************************/
if feedback_flag = 'OFF'
  then return
Address CPSM 'QUERY OBJECT(PROGRAM)',
             'THREAD(w_thread)',
             'RESULT(w_result)',
             'DATALENGTH(w_into_objectlen)',
             'RESPONSE(w_response)',
             'REASON(w_reason)'
if w_response <> EYURESP(OK)
  then
    do
      msg = 'Failure QUERYing OBJECT(PROGRAM):'
      say rexx_name msg
      retc = 16
      msg = 'Response = 'EYURESP(w_response),
            'Reason = 'EYUREAS(w_reason)
      say rexx_name msg
      Signal CPSM_DISCONNECT
    end
msg = 'Each OBJECT(PROGRAM) record is 'w_into_objectlen' bytes.'
say rexx_name msg

/*********************************************************************/
/*  Loop through the PROGRAM results and the FEEDBACK records and    */
/*  translate the output into displayable characters. Issue a        */
/*  message for each PROGRAM result with the appropriate FEEDBACK.   */
/*  If CICSPlex SM commands receive a response other than OK then    */
/*  issue diagnostic messages and set the return code before         */
/*  disconnecting and terminating. Only performed if FEEDBACK        */
/*  records are present.                                             */
/*********************************************************************/
if feedback_flag = 'OFF'
  then return
do iii = 1 to feedback_count
  Address CPSM 'TPARSE OBJECT(FEEDBACK)',
               'PREFIX(fdb)',
               'STATUS(w_response)',
               'VAR(feedback.iii)',
               'THREAD(w_thread)'
  if w_response <> 'OK'
    then
      do
        msg = 'Failure parsing OBJECT(FEEDBACK). Status='w_response
        say rexx_name msg
        retc = 16
        Signal CPSM_DISCONNECT
      end
  Address CPSM 'FETCH INTO(w_into_object)',
               'LENGTH(w_into_objectlen)',
               'POSITION('fdb_rsltrecid')',
               'RESULT(w_result)',
               'THREAD(w_thread)',
               'RESPONSE(w_response)',
               'REASON(w_reason)'
  if w_response <> EYURESP(OK)
    then
      do
        msg = 'Failure FETCHing record:'
        say rexx_name msg
        retc = 16
        msg = 'Response = 'EYURESP(w_response),
              'Reason = 'EYUREAS(w_reason)
        say rexx_name msg
        Signal CPSM_DISCONNECT
      end
  Address CPSM 'TPARSE OBJECT(PROGRAM)',
               'PREFIX(program)',
               'STATUS(w_response)',
               'VAR(w_into_object.1)',
               'THREAD(w_thread)'

  if w_response <> 'OK'
    then
      do
        msg = 'Failure parsing OBJECT(PROGRAM). Status='w_response
        say rexx_name msg
        retc = 16
        Signal CPSM_DISCONNECT
      end
  /*******************************************************************/
  /*  Format the display information for the PROGRAM results.        */
  /*  cicsname PROGRAM(        ) LANG(            )                  */
  /*           LEN(          ) DFHRPL(  ) FN(        )               */
  /*           RESP(    ) RESP2(    )                                */
  /*******************************************************************/
  msg = '   'program_eyu_cicsname' PROGRAM(        )',
        'LANG(           ) LEN(ØØØØØØØØØØ) DFHRPL(ØØ) FN(ØØØØØØØØ)',
        ,RESP(ØØØØ) RESP2(ØØØØ)'
  msg = 'OVERLAY'(program_program,msg,21)
  msg = 'OVERLAY'(program_language,msg,36)
  len_pos = 64 - 'LENGTH'(program_length)
  msg = 'OVERLAY'(program_length,msg,len_pos)
  rpl_pos = 75 - 'LENGTH'(program_rplid)
  msg = 'OVERLAY'(program_rplid,msg,rpl_pos)
```

```
  fn_pos = 88 - 'LENGTH'(fdb_ceibfn)
  msg = 'OVERLAY'(fdb_ceibfn,msg,fn_pos)
  resp_pos = 99 - 'LENGTH'(fdb_ceibresp)
  msg = 'OVERLAY'(fdb_ceibresp,msg,resp_pos)
  resp2_pos = 111 - 'LENGTH'(fdb_ceibresp1)
  msg = 'OVERLAY'(fdb_ceibresp1,msg,resp2_pos)
  say rexx_name msg
end iii
CPSM_DISCONNECT:
/*******************************************************************/
/*  Call external subroutine CPSMDISC to disconnect from CICSPlex SM */
/*  using the CICSPlex SM token "w-thread". Regardless of the        */
/*  return code from CPSMDISC termination processing continues       */
/*  normally.                                                        */
/*******************************************************************/
Call CPSMDISC w_thread
/*******************************************************************/
/*  Call external subroutine CPSMTERM to terminate the CICSPlex SM   */
/*  REXX/API environment. Regardless of the return code from         */
/*  CPSMTERM termination processing continues normally.              */
/*******************************************************************/
Call CPSMTERM
RETURN_CONTROL:
/*******************************************************************/
/*  Return control with a return code - this is the only exit point  */
/*******************************************************************/
msg = 'Terminated with return code: 'retc
say rexx_name msg
exit (retc)
```

The following example performs a PHASEIN for all application programs in the CICS region CICSTA01 within the CICSplex TPLEX:

```
CPSMAPGM TPLEX CICSTAØ1

CPSMAPGM Invoked on 18 Nov 2ØØ2 at 1Ø:45:55.
CPSMAPGM Invoked with parameters:-
CPSMAPGM Context  : TPLEX
CPSMAPGM Scope    : CICSTAØ1
CPSMAPGM Criteria : NOT (PROGRAM=C* OR PROGRAM=DBUG* OR PROGRAM=DFH* OR
PROGRAM=DSN* OR PROGRAM=EYU* OR PROGRAM=IBM* OR PROGRAM=IGZ*
OR HOLDSTATUS=HOLD)
CPSMAPGM PERFORMed ACTION(PHASEIN) for 34 programs.
CPSMAPGM ACTION(PHASEIN) successful for 32 programs.
CPSMAPGM ACTION(PHASEIN) failed for 2 programs.
CPSMAPGM Each OBJECT(PROGRAM) record is 4Ø8 bytes.
CPSMAPGM    CICSTAØ1 PROGRAM(XVTTAHØS) LANG(ASSEMBLER   )
LEN(ØØØØØØØØØØ) DFHRPL(ØØ) FN(ØØØØ4EØ4) RESP(ØØ17) RESP2(ØØØ8)
```

```
CPSMAPGM    CICSTAØ1 PROGRAM(XVTTARLS) LANG(ASSEMBLER   )
LEN(ØØØØØØØØØØ) DFHRPL(ØØ) FN(ØØØØ4EØ4) RESP(ØØ17) RESP2(ØØØ8)
CPSMAPGM Terminated with return code: Ø
```

Carl Wade McBurnie
IT Consultant (Germany)

CICS news

CONNX Solutions has announced Version 8.8 of its CONNX data access middleware, now with a range of performance and feature enhancements. Support for VSAM VSE data sources, which provides real-time high-performance access to VSAM files under CICS partitions on the VSE operating system, has also been included in the release. Also, direct support for Microsoft .NET technology has been added with the introduction of a pure CONNX OLE DB Provider. Users, says the vendor, will be able to achieve the performance of a native provider while writing their own applications in managed C# or VB .NET code.

The new release also includes a new product called the CONNX Data Synchronization Tool, which allows users to move enterprise data from any source location to any target data source. The Windows-based tool uses a technique of hash keys to detect when records are updated, deleted, or inserted.

By storing these hash keys, along with the primary key of each record, in a separate CONNX private data store, it performs incremental updates from source to target at, it's claimed, a fraction of the usual full update speed.

This is apparently suited for creating and maintaining data warehouses and data marts. Once the target source is populated, it can be used with other tools for *ad hoc* reporting, application development, and Web development. The engine provides the flexibility of querying either against the data warehouse and data marts or against the live operational data sources.

It acts as a reusable data access framework, supporting C-ISAM, VSAM, DB2, Oracle, RMS, RDB, PostGreSQL, DBMS, Dataflex, POWERflex, SQL Server, Sybase, and Informix and any OLE DB, ODBC, or JDBC data source.

For further information contact:
CONNX Solutions, 1800 112th Avenue NE, Suite #150, Bellevue, WA 98004, USA.
Tel: (425) 519 6600.
URL: http://www.connx.com/products/products.html.

* * *

Neon Systems has announced general availability of its Shadow Console interleaved management, monitoring, and debugging tool, designed to ensure the performance and availability of composite applications that integrate application platform suites with mainframe data and transactions.

Through its Windows-based GUI, it promises to offer application developers and production operations staff a consolidated end-to-end view of the middleware component from the initial API adapter call in the application platform suite to the back-end data or transaction source on the mainframe.

The software can be deployed with leading J2EE and .NET application platform suites to provide customers with J2CA, JDBC, or ODBC access to mainframe data sources and transaction environments, supporting DB2, CICS/TS, IMS/TM, IMS/DB, VSAM, ADABAS, Natural/ACI, flat files, IDMS, and other z/OS data and transactional sources.

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