February 2003

In this issue

3  Communication between batch and CICS
9  CICSPlex SM API – REXX EXECs (run-time interface)
29  Monitoring CICS resources online – part 2
38  CICSPlex SM dynamic workload management – workloads
50  Book review – Murach’s CICS Desk Reference
51  CICS questions and answers
53  CICS news

© Xephon plc 2003
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.

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.
Communication between batch and CICS

It is often necessary to communicate between a CICS transaction and a batch process, or for a CICS transaction to launch a batch process (a JOB).

It is possible to link a batch program with a CICS program by using the EXCI interface. However, if the process is long or costly, it is preferable to launch a CICS transaction from batch, and to separate both processes.

It is also possible to launch a batch JOB from a CICS program. This makes it possible to execute a process with a lower priority in batch and avoid penalizing the general response times of CICS.

LAUNCH A CICS TRANSACTION FROM BATCH

To launch a CICS transaction from batch, the JCL must include the following MVS commands:

```mvs
//BTCHCICS JOB (Ø,Ø), .......
//*
//PAS EXEC PGM=IEFBR14
//  COMMAND 'F CICSX,TTTT'
//*
```

In this example, the command is executed immediately by JES2, without waiting for the completion of PAS. An EXEC is necessary to avoid the message “JOB NOT RUN - JCL ERROR”. In the command, TTTT it is the transaction to execute in CICSX. The command must be placed between quotes.

So that the command is executed to the completion of a step, it must be written in the following form:

```mvs
//JOBBTCH JOB .......
//PAS1 EXEC ......
//  ........
//PAS2 EXEC ....
//  ........
//PAS3 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=X
```
Thus the JOB BTCHCICS is sent to the internal reader in PAS3 and is executed after PAS2.

So that CICS accepts commands from batch, one must define the terminal CJCL that supplies CICS in the group DFH$CNSL of DFHCSD.

If the transaction has restricted access to a given user, ie it requires a SIGN-ON, the following will be added as the first command with the transaction CESN:

```
//BTCHCICS JOB ……
//PAS EXEC PGM=IEFBR14
//COMMAND 'F CICSX,CESN USERID=UUUUUUUU,PS=PPPPPPPPP'
//COMMAND 'F CICSX,TTTT'
```

To avoid having the password appear in the JCL, it can be coded using only the userid:

```
// COMMAND 'F CICSX, CESN USERID=UUUUUUUU'
```

CICS will ask the operator to enter the password on the console:

```
@NN DFHCE3523 CICSX PLEASE TYPE YOUR PASSWORD.
```

The answer on the console would be the password, ie:

```
NN PPPPPPPPP
```

It is also possible to assign a userid to the terminal CJCL.

Data can be sent to the CICS program from batch by adding the following command:

```
// COMMAND 'F CICSX,TTTT,DATA TO RECEIVE BY THE PROGRAM'
```

The program CICS will use is:

```
EXEC CICS RECEIVE INTO(W-DATOS) LENGTH(W-LONG) END-EXEC
```

and W-DATOS is defined, for example, by:
The program can send a message (OS/390 SYSLOG) to the terminal CJCL, for example:

```plaintext
       01 W-MESSAGE PIC X(22)
          VALUE "TRANS TTTT STARTED OK."
```

MOVE 22 TO W-LONG
EXEC CICS SEND TEXT FROM(W-MESSAGE) LENGTH(W-LONG)
TERMINAL FREEKB ALARM ERASE END-EXEC

Another transaction that can be sent from batch by a command is the CICS transaction CMSG, which sends a message from a JOB to one terminal or to several terminals.

An example is to notify a terminal of the successful completion of a batch process:

```plaintext
// JOBBTCH JOB ....
//       ....
// IFOK IF RC = 0 THEN
//PASOK EXEC PGM=IEBGENER
// SYSPRINT DD SYSOUT=X
// SYSUT1 DD DATA, DLM=@@
//*
// BTCHCICS JOB ....
// PAS EXEC PGM=IEFBR14
// COMMAND 'F CICSX, CMSG MSG="JOB JOBBTCH OK"', R=TERM, SEND'
@@
// SYSIN DD DUMMY
// SYSUT2 DD SYSOUT=(A,INTRDR)
// IFOKEND ENDIF
```

The CICS reply is:

```
+M R S OK MESSAGE HAS BEEN ROUTED
```

Error messages that occur most frequently are described below.

If the CICS does not exist:

```
IEE3411 CI CSX NOT ACTIVE
```

If the transaction does not exist:
DFHAC2001  CICSX TRANSACTION 'TTTT' IS NOT RECOGNIZED. CHECK THAT THE
TRANSACTION NAME IS CORRECT.

If the transaction is disabled:

DFHAC2008  CICSX TRANSACTION TTTT HAS BEEN DISABLED AND CANNOT BE USED.

LAUNCH A BATCH JOB FROM CICS

To launch a batch JOB from CICS one must define a TD QUEUE EXTRAPARTITION and write to it the JCL to execute in JES2.

The definition of a TD QUEUE is:

CEDA  View TDqueue( BTCH )
    TDqueue        : BTCH
    Group          : DCTGEN
    Description    :
    TYPE           : Extra

EXTRA PARTITION PARAMETERS
    Databuffers    : 001
    DDname         :
    DSname         :
    Sysoutclass    :
    Erroroption    : Ignore
    Opentime       : Initial
    REWind         : Leave
    TYPEFile       : Output
    RECORDSize     : 000000
    BLOCKSize      : 000000
    RECORDFormat   : Fixed
    BLOCKFormat    : Blocked
    Printcontrol   :
    Disposition    : Shr

INTRA PARTITION PARAMETERS

......

Add in the PROCEDURE to start in CICS a DD with a name equal to the parameter DDNAME of the defined TDQUEUE:

    //JOBBTCH  DD SYSOUT=(A, INTRDR),
    //           DCB=(RECFM=FB, LRECL=80, BLKSIZE=80)

To make it more dynamic to launch a batch JOB, it is preferable to launch a standard PROCEDURE that reads a member of a library with the JCL to execute. In this way the programs contain the name of the member to launch and this avoids having to modify the programs after each change to the JCL.
The CICS program that writes in the TDQUEUE will be more or less as shown below.

In the WORKING STORAGE SECTION put:

```
03 W-JCLJOB       PIC X(80) VALUE '//'JLAUNCH JOB (0,0),CICS BATCH,CLASS=A,MSGCLASS=Y'.
03 W-JCLEXEC      PIC X(80) VALUE '//'PAS EXEC PGM=IEFBR14'.
03 W-JCLSTRT.
05 FILLER        PIC X(14) VALUE ' S PLAUNCH,J='. 
05 W-MEMBER      PIC X(8) VALUE SPACES.
05 FILLER        PIC X(58) VALUE SPACES.
03 W-JCLEOF      PIC X(80) VALUE '/*EOF'.
```

In the PROCEDURE DIVISION put:

```
MOVE +80 TO W-LONG.
MOVE 'name of member' TO W-MEMBER.
EXEC CICS WRITEQ TD QUEUE(W-CUA) FROM(W-JCLJOB) LENGTH(W-LONG) END-EXEC.
EXEC CICS WRITEQ TD QUEUE(W-CUA) FROM(W-JCLEXEC) LENGTH(W-LONG) END-EXEC.
EXEC CICS WRITEQ TD QUEUE(W-CUA) FROM(W-ARRANQUE) LENGTH(W-LONG) END-EXEC.
EXEC CICS WRITEQ TD QUEUE(W-CUA) FROM(W-JCLEOF) LENGTH(W-LONG) END-EXEC.
EXEC CICS WRITEQ TD QUEUE(W-CUA) FROM(W-JCLEOF) LENGTH(W-LONG) END-EXEC.
```

This will launch a batch JOB JLAUNCH, which starts the PROC PLAUNCH. The command with the data W-JCLEOF is sent twice to force the execution of the JOB in the internal reader immediately.

The PROC PLAUNCH is:

```
//PROD PROC J=MEMBER
//PAS01 EXEC PGM=IEBGENER
//SYSPRINT DD SYSOUT=Y
//SYSUT1 DD DSN=JCL.LIBRARY.DATA(&J),DISP=SHR
//SYSUT2 DD DUMMY
//SYSUT2 DD SYSOUT=(A,INTRDR)
```

where JCL.LIBRARY.DATA is the dataset where each member is a JOB.

In the OS/390 SYSLOG the following messages will appear:
The JOB in member is executed afterwards.

Juan Tormo  
Systems Manager  
Sidmed SA (Spain)  © Xephon 2003

In addition to CICS Update, the Xephon family of Update publications now includes AIX Update, DB2 Update, MQ Update, MVS Update, TCP/SNA Update, and RACF Update. Although the articles published are of a very high standard, the vast majority are not written by professional writers, and we rely heavily on our readers themselves taking the time and trouble to share their experiences with others. Many have discovered that writing an article is not the daunting task that it might appear to be at first glance.

They have found that the effort needed to pass on valuable information to others is more than offset by our generous terms and conditions and the recognition they gain from their fellow professionals. Often, just a few hundred words are sufficient to describe a problem and the steps taken to solve it.

If you have ever experienced any difficulties with CICS, or made an interesting discovery, you could receive a cash payment, a free subscription to any of our Updates, or a credit against any of Xephon’s wide range of products and services, simply by telling us all about it. For a copy of our Notes for Contributors, which explains the terms and conditions under which we publish articles, please point your browser at www.xephon.com/nfc. Articles can be sent to the editor, Trevor Eddolls, at trevore@xephon.com.
INTRODUCTION

The CICSPlex SM Applications Programming Interface (API) is an exceptionally versatile interface for the management of CICS regions, CICS resources, and CICSPlex SM itself.

What makes the CICSPlex SM API so versatile and flexible is the ability to specify a context, a scope within the context, and criteria within the scope.

The context is the name of a CICSPlex SM Address Space (CMAS) or CICSpex.

If the context is a CICSpex, the scope further qualifies the context by specifying that the scope is the CICSpex itself, a CICS system group, or a specific CICS system. If the context is a CMAS, the scope has no meaning and is ignored.

The criteria option enables the filtering of resource tables using simple expressions, eg for a PROGRAM resource:

```
PROGRAM=PROG1.
```

and complex, compound logical expressions, eg for a LOCTRAN resource:

```
(TRANID=P* AND PROGRAM=PROG1 AND STATUS=ENABLED) AND
((USECOUNT>0 AND STGVCNT>0) OR NOT RESTARTCNT=0).
```

See *CICSPlex SM Resource Tables Reference* for details of the possible attributes for each resource table. The attributes are obtained from a number of sources – CICSPlex SM services, CICS Systems Programming Interface (SPI) INQUIRE and STATISTICS, and also CICS Monitoring Facility (CMF) performance class records.

The ability to specify filter expressions is an enormous improvement when compared with the CICS SPI, where for
example you have to ‘browse’ through all the resource table entries, compare the fields yourself, and, when you have found what you are looking for, issue a SET command. With the CICSPlex SM API you can select based on your criteria, and process the required action in a single command. This makes programming with the CICSPlex SM API simpler and also enables you to easily write very ‘open’ and flexible programs.

The CICSPlex SM API has two interfaces – a command-level interface for programs written in Assembler, PL/I, COBOL, or C, and a run-time interface, which supports programs written as REXX EXECs.

This article concentrates on programs written as REXX EXECs for the run-time interface.

THE RUN-TIME INTERFACE

The CICSPlex SM API run-time interface is supplied with CICSPlex SM as a REXX function package and a host command. For information about installing the REXX function package and host command environment see CICS Transaction Server for OS/390: Installation Guide.

There was an article in CICS Update in January 2002 by Dr Paul Johnson about the CICSPlex SM API run-time interface (CICSPlex SM API program written in REXX) and there is also a CICS SupportPac CS13 – CICSPlex SM Sample API by Iain Coles.

One advantage of the run-time interface is that it is ideal for experimenting with the CICSPlex SM API – you can write your REXX and execute it immediately for immediate results. Some of the REXX EXECs in this article were later re-written as Assembler programs, so they were, to a certain extent, a development ‘stepping stone’ to test that the CICSPlex SM API commands were achieving the desired results.

THE REXX EXECs

There are four external subroutines, which are used by all the
REXX EXECs:

- CPSMINIT – initialize the CICSPlex SM API environment
- CPSMCONN – connect to CICSPlex SM
- CPSMDISC – disconnect from CICSPlex SM
- CPSMTERM – terminate the CICSPlex SM API environment.

There are seven main REXX EXECs:

- CPSMTIME – performs RESETTIME for CICS regions
- CPSMTASK – displays CICS task information
- CPSMTRNC – displays CICS transaction class information
- CPSMTSQI – displays CICS temporary storage queue (TSQ) information
- CPSMTSQD – deletes CICS TSQs
- CPSMNEWC – performs CICS program PHASEIN
- CPSMAPGM – performs CICS program PHASEIN for all ‘application’.

### CPSMINIT

```rexx
/****************************** REXX **********************************/
/*  MODULE NAME : CPSMINIT                                           */
/*  MODULE TYPE : REXX Executable (external subroutine)              */
/*  DESCRIPTION : CICSPlex SM - Initialize REXX/API                  */
/*               Initializes the CICSPlex SM REXX/API environment.     */
/*  INVOCATION  : CPSMINIT                                           */
/*  RETURN      : Parameters returned in the variable "result":       */
/*                retc      = return code                            */
/*********************************************************************/
Trace
rexx_name = 'CPSMINIT'
retc = Ø
api = EYUINIT()
if api <> Ø then do
   msg = 'Failure initializing CPSM REXX/API environment. RC='api
```
CPSMCONN

CPSMCONN was written for the release of CICSPlex SM supplied as a component of CICS Transaction Server for OS/390 Version 1. If you have a later release of CICSPlex SM and require access to additional resource table attributes available in that later release, you will have to change the VERSION option of the CONNECT command in CPSMCONN.

```rexx
say rexx_name msg
retc = 16
end
return retc
```

CPSMCONN

CPSMCONN was written for the release of CICSPlex SM supplied as a component of CICS Transaction Server for OS/390 Version 1. If you have a later release of CICSPlex SM and require access to additional resource table attributes available in that later release, you will have to change the VERSION option of the CONNECT command in CPSMCONN.

```rexx
ifndef module
module name = CPSMCONN
module type = REXX executable (external subroutine)
description = CICSPlex SM - CONNECT
invocation = CPSMCONN 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 = Parameters returned in the variable "result":
         retc = return code
         w_thread = CICSPlex SM token
endif
```

```rexx
Trace
rexx_name = 'CPSMCONN'
retc = 0
Parse Upper Arg w_context, w_scope

Address CPSM 'CONNECT CONTEXT('w_context')',
       'SCOPE('w_scope')',
       'THREAD(w_thread)',
       'VERSION(Ø14Ø)',
       'RESPONSE(w_response)',
       'REASON(w_reason)'
if w_response <> EYURESP(OK)
then
do
  msg = 'Failure CONNECTing to CPSM:'
say rexx_name msg
  retc = 16
  msg = 'Response = 'EYURESP(w_response),
```
CPSMDISC

/******************************************************************************
/* MODULE NAME : CPSMDISC */
/* MODULE TYPE : REXX Executable (external subroutine) */
/* DESCRIPTION : CICSPlex SM - DISCONNECT */
/* Disconnects from CICSPlex SM using the thread specified at invocation. */
/* INVOCATION : CPSMDISC w_thread */
/* w_thread = CICSPlex SM token */
/* RETURN : Parameters returned in the variable "result":- */
/* retc = return code */
/******************************************************************************

Trace
rexx_name = 'CPSMDISC'
retc = Ø

Parse Upper Arg w_thread
Address CPSM 'DISCONNECT THREAD(w_thread)',
'RESPONSE(w_response)',
'REASON(w_reason)'
if w_response <> EYURESP(OK)
then
do
  msg = 'Failure DISCONNECTing from CPSM:'
say rexx_name msg
  retc = 16
  msg = 'Response = 'EYURESP(w_response),
       'Reason = 'EYUREAS(w_reason)
say rexx_name msg
end
return retc

CPSMTERM

/******************************************************************************
/* MODULE NAME : CPSMTERM */
/* MODULE TYPE : REXX Executable (external subroutine) */
/* DESCRIPTION : CICSPlex SM - Terminate REXX/API */
/* Terminates the CICSPlex SM REXX/API environment. */
/* INVOCATION : CPSMTERM */
/* RETURN : Parameters returned in the variable "result":- */
/* retc = return code */
/******************************************************************************

CPSMTIME

CPSMTIME does not use criteria, it simply processes the EXEC CICS PERFORM RESETTIME command for the CICS regions within the context and scope specified.

```rexx
/**********************************************************************************/
Trace
rexx_name = 'CPSMTIME'
retc = Ø
w_context = ''
w_scope = ''
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.'
   end
return retc
/**********************************************************************************/
```

```rexx
/**********************************************************************************/
/* MODULE NAME : CPSMTIME */
/* MODULE TYPE : REXX Executable */
/* DESCRIPTION : CICSPlex SM - RESETTIME */
/* Synchronizes the CICS date and time-of-day with */
/* the system date and time-of-day for the CICS */
/* regions within the context and scope specified. */
/* This is particularly useful for the transition */
/* from summer-time (daylight saving time) to */
/* winter-time and vice versa. */
/* INVOCATION : CPSMTIME 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 */
/**********************************************************************************/
```
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
/*********************************************************************/
/* 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(RESETTIME) for the OBJECT(CICSRGN) with no */
/* additional parameter requirements. If the response code from */
/* CICSPlex SM is not OK then issue diagnostic messages and set */
/* return code before disconnecting and terminating. If the */
/* response code is OK then issue a message with the number of */
/* CICS regions affected. */
/*********************************************************************/
Address CPSM 'PERFORM OBJECT(CICSRGN)',
'ACTION(RESETTIME)',
'COUNT(w_count)',
if w_response <> EYURESP(OK)
then
do
  msg = 'Failure PERFORMing ACTION(RESETTIME):'
  say rexx_name msg
  retc = 16
  msg = 'Response = 'EYURESP(w_response),
       'Reason = 'EYUREAS(w_reason)
  say rexx_name msg
  Signal CPSM_DISCONNECT
end
msg = 'PERFORMed ACTION(RESETTIME) for 'w_count' CICS regions.'
say rexx_name msg

Address CPSM 'QUERY OBJECT(CICSRGN)',
  'RESULT(w_result)',
  'DATALENGTH(w_into_objectlen)',
  'RESPONSE(w_response)',
  'REASON(w_reason)'
if w_response <> EYURESP(OK)
then
do
  msg = 'Failure QUERYing OBJECT(CICSRGN):'
  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(CICSRGN) record is 'w_into_objectlen' bytes.'
say rexx_name msg

Address CPSM 'QUERY OBJECT(CICSRGN)',
  'RESULT(w_result)',
  'DATALENGTH(w_into_objectlen)',
  'RESPONSE(w_response)',
  'REASON(w_reason)'
if w_response <> EYURESP(OK)
then
do
  msg = 'Failure QUERYing OBJECT(CICSRGN):'
  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(CICSRGN) record is 'w_into_objectlen' bytes.'
say rexx_name msg

*********************************************************************/
/* Obtain information about the CICSRGN object, i.e. 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(CICSRGN)',
  'RESULT(w_result)',
  'DATALENGTH(w_into_objectlen)',
  'RESPONSE(w_response)',
  'REASON(w_reason)'
if w_response <> EYURESP(OK)
then
do
  msg = 'Failure QUERYing OBJECT(CICSRGN):'
  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(CICSRGN) record is 'w_into_objectlen' bytes.'
say rexx_name msg

/***************************************************************************/
/* Loop through the CICSRGN results and translate the output */
/* into displayable characters. Issue a message for each CICS */
/* region affected. If the 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(CICSRGN)',
    'PREFIX(cicsrgn)',
    '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 
end

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 RESETTIME for all CICS regions in the CICS system group TEST in CICSpex TPLEX:

CPSMTIME TPLEX TEST

CPSMTIME Invoked on 18 Nov 2002 at 09:43:59.
CPSMTIME Invoked with parameters:·
CPSMTIME Context : TPLEX
CPSMTIME Scope    : TEST
CPSMTIME PERFORMed ACTION(RESETTIME) for 3 CICS regions.
CPSMTIME Each OBJECT(CICSRGN) record is 880 bytes.
CPSMTIME  CICSTA01 JOBNAME(CICSTA01) APPLID(APPLTA01) MVS(TEST)
CPSMTIME  CICSTA02 JOBNAME(CICSTA02) APPLID(APPLTA02) MVS(TEST)
CPSMTIME  CICSTT01 JOBNAME(CICSTT01) APPLID(APPLTT01) MVS(TEST)
CPSMTIME Terminated with return code: 0

CPSMTASK

CPSMTASK displays 'basic' information about CICS tasks based on the context, scope, and criteria specified. The CICSpex SM TASK resource table has a large number of attributes that could be used as criteria. The CPSMTASK example shows that there are no less than 1,536 bytes of information available for each active task.

/***************************************************************************/
/* MODULE NAME : CPSMTASK                                               */
/* MODULE TYPE : REXX Executable                                       */
/* DESCRIPTION : CICSpex SM - TASK Resource Information                */
/***************************************************************************/
/* Displays information about CICS tasks based on the parameters provided. */
/* INVOCA TION: CPSMTASK 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 TASKs selected */
/* RETURN : retc = return code */
/* ***********************************************/

Trace
rexx_name = 'CPSMTASK'
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 TASK objects based on the context, scope, and additional */
/* 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 because this response indicates that no */
/* tasks matched the criteria specified within the context and */
/* scope. Issue a message that zero records were retrieved, */
/* disconnect, and terminate. If the response is OK then process */
/* the task records. */
*******************************************************************************/

w_criteria = w_criteria||"."
w_criteriaLen = 'LENGTH'(w_criteria)
Address CPSM 'GET OBJECT(TASK)',
  '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) & w_response <> EYURESP(NODATA) then
  do
    msg = 'Failure GETting TASK:'
    say rexx_name msg
    retc = 16
    msg = 'Response = 'EYURESP(w_response),
      'Reason = 'EYUREAS(w_reason)
    say rexx_name msg
    Signal CPSM_DISCONNECT
  end
  msg = 'GET retrieved 'w_count' OBJECT(TASK) records.'
  say rexx_name msg
  if w_response = EYURESP(NODATA) then Signal CPSM_DISCONNECT
/* Obtain information about the TASK 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(TASK)',
   '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(TASK):'
      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(TASK) record is 'w_into_objectlen' bytes.'
say rexx_name msg
*********************************************************************/
/* Loop through the TASK results and translate the output into */
/* displayable characters. Issue a message for each CICS task. 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)',
      '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(Task)',
  'PREFIX(task)',
  '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 TASK results. */
/*******************************************************************/
msg = '   ' task_eyu_cicsname ' TASK(ØØØØØØØ) TRAN(    ) USER(        ) TERM(    ) st ' 
pos = 25 - 'LENGTH'(task_task)
msg = 'OVERLAY'(task_task, msg, pos)
msg = 'OVERLAY'(task_tranid, msg, 32)
msg = 'OVERLAY'(task_userid, msg, 43)
msg = 'OVERLAY'(task_termid, msg, 58)
msg = 'OVERLAY'(task_runstatus, msg, 64, 3)
say rexx_name msg
end
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)
CPSMTASK could, for example, be used to display all CICS tasks with a specific or generic transaction identifier, or user identifier, or priority, or any combination of those and many other attributes.

The following example displays all CICS tasks in the CICS system group TEST within the CICSp lex TPLEX with a transaction identifier beginning with ‘C’ and which are suspended because of MQSeries:

CPSMTASK TPLEX TEST TRANID=C* AND SUSPENDTYPE=MQS*

CPSMTASK Invoked with parameters:
CPSMTASK Context : TPLEX
CPSMTASK Scope : TEST
CPSMTASK Criteria : TRANID=C* AND SUSPENDTYPE=MQS*
CPSMTASK GET retrieved 2 OBJECT(TASK) records.
CPSMTASK Each OBJECT(TASK) record is 1536 bytes.
CPSMTASK CICSTA01 TASK(0000031) TRAN(CKTI) USER(CICSRGN) TERM( ) SUS
CPSMTASK CICSTA02 TASK(0000029) TRAN(CKTI) USER(CICSRGN) TERM( ) SUS
CPSMTASK Terminated with return code: 0

CPSMTRNC

CPSMTRNC displays information about CICS transaction classes based on the context, scope, and criteria specified. The CICSp lex SM TRANCLAS resource table has a number of attributes that could be used as criteria. The CPSMTRNC example shows that there are 104 bytes of information available for each transaction class instance.

If you are using CICS TS 1.3 ensure that the PTF UQ61178 for APAR PQ55708 is applied.

/**************************** REXX *******************************/
/*                      MODULE NAME : CPSMTRNC                     */
/*                      MODULE TYPE : REXX Executable              */
/*                      DESCRIPTION : CICSp lex SM - TRANCLAS Resource Information */
/*                      Displays information about TRANCLAS resources */
/*                      based on the parameters provided.           */
/*                      INVOCATION : CPSMTRNC w_context w_scope w_criteria */
/*                        w_context = name of a CMAS or CICSp lex           */
/*                        w_scope = name of CICSp lex, CICS system group    */
/*                        or CICS system within w_context                 */
/*                w_criteria= the criteria to be used to filter      */
/*                            the TRANCLAS objects                   */
/*  RETURN      : retc      = return code                            */
/*********************************************************************/
Trace
rexx_name = 'CPSMTRNC'
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     */
/*  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

© 2003. Xephon UK telephone 01635 33848, fax 01635 38345. USA telephone (303) 410 9344, fax (303) 438 0290.
Parse Var result w_retc w_thread.
if w_retc <> 0
 then
do
  retc = w_retc
  Call CPSMTERM
  Signal RETURN_CONTROL
end

/*********************************************************************/
/* GET the TRANCLAS 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       */
/* transaction class objects matched the criteria specified within */
/* the context and scope. Issue a message that zero records were   */
/* retrieved, disconnect and terminate. If the response is OK then  */
/* process the transaction class records.                          */
/*********************************************************************/

w_criteria = w_criteria||"."
w_criteriaen = 'LENGTH'(w_criteria)
Address CPSM 'GET OBJECT(TRANCLAS)',
   'CRITERIA(w_criteria)',
   'LENGTH'(w_criteriaen)',
   'COUNT(w_count)',
   'RESULT(w_result)',
   'THREAD(w_thread)',
   'RESPONSE(w_response)',
   'REASON(w_reason)'
if w_response <> EYURESP(OK) & w_response <> EYURESP(NODATA)
 then
do
  msg = 'Failure GETting TRANCLAS:'
say rexx_name msg
  retc = 16
  msg = 'Response = 'EYURESP(w_response),
       'Reason = 'EYUREAS(w_reason)
say rexx_name msg
  Signal CPSM_DISCONNECT
end
msg = 'GET retrieved 'w_count' OBJECT(TRANCLAS) records.'
say rexx_name msg
if w_response = EYURESP(NODATA)
 then Signal CPSM_DISCONNECT

/*****************************************************************************/
/* Obtain information about the TRANCLAS 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   */
Address CPSM 'QUERY OBJECT(TRANCLAS)',
'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(TRANCLAS):
    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(TRANCLAS) record is 'w_into_objectlen' bytes.'
say rexx_name msg
  /* Loop through the TRANCLAS results and translate the output into displayable characters. Issue a message for each TRANCLAS. 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)',
    '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(TRANCLAS)',
  'PREFIX(tcl)',
  'STATUS(w_response)',
  'VAR(w_into_object.1)'


if w_response <> 'OK'
then
do
    msg = 'Failure parsing record. Status=' w_response
    say rexx_name msg
    retc = 16
    Signal CPSM_DISCONNECT
end

CPSM_DISCONNECT:
Call CPSMDISC w_thread
CPSMTERM:
RETURN_CONTROL:
msg = 'Terminated with return code: ' retc
say rexx_name msg
exit (retc)

CPSMTRNC could, for example, be used to display all CICS transaction class instances where transactions were queued or
where transactions had to be purged, or any combination of attributes.

The following example displays all CICS transaction class instances in the CICSTT01 region within the CICSplex TPLEX with a transaction class name that does not begin with ‘DFH’.

```
CPSMTRNC TPLEX CICSTT01 NAME^=DFH*

CPSMTRNC Invoked on 18 Nov 2002 at 10:22:42.
CPSMTRNC Invoked with parameters:
CPSMTRNC Context : TPLEX
CPSMTRNC Scope : CICSTT01
CPSMTRNC Criteria : NAME^=DFH*
CPSMTRNC GET retrieved 10 OBJECT(TRANCLAS) records.
CPSMTRNC Each OBJECT(TRANCLAS) record is 104 bytes.
CPSMTRNC CICSTT01 TCL(TCLI) MAX(0015) ACT(0000) PUR(0000)
CPSMTRNC CICSTT01 TCL(TCLP) MAX(0005) ACT(0000) PUR(0000)
CPSMTRNC CICSTT01 TCL(TCLQ) MAX(0015) ACT(0000) PUR(0000)
CPSMTRNC CICSTT01 TCL(TCLU) MAX(0010) ACT(0000) PUR(0000)
CPSMTRNC CICSTT01 TCL(TCLX) MAX(0020) ACT(0000) PUR(0000)
CPSMTRNC CICSTT01 TCL(TCL1) MAX(0015) ACT(0000) PUR(0000)
CPSMTRNC CICSTT01 TCL(TCL2) MAX(0005) ACT(0000) PUR(0000)
CPSMTRNC CICSTT01 TCL(TCL3) MAX(0015) ACT(0000) PUR(0000)
CPSMTRNC CICSTT01 TCL(TCL4) MAX(0010) ACT(0000) PUR(0000)
CPSMTRNC CICSTT01 TCL(TCL5) MAX(0020) ACT(0000) PUR(0000)
CPSMTRNC Terminated with return code: 0
```

Editor’s note: this article will be concluded in the next issue.

Carl Wade McBurnie

IT Consultant (Germany) © Xephon 2003
Monitoring CICS resources online – part 2

This month we conclude the code that allows users to check on the status of various resources in their CICS systems and send messages to the appropriate people when the resources are not available.

```Assembly
WHEN DDNAM
  IF CURR-OBS = DFHVALUE(CLOSED) AND
     LAST-OBS(I) = DFHVALUE(CLOSED)
  PERFORM P210Ø-BUILD-MSG THRU P210Ø-BUILD-MSG-EXIT
  ELSE MOVE CURR-OBS TO LAST-OBS(I)
  PERFORM P220Ø-OK THRU P220Ø-OK-EXIT
END-IF
WHEN OTHER
  IF LAST-OBS(I) = CURR-OBS AND
     LONG-RUNNING-FLAG = ZERO
  PERFORM P210Ø-BUILD-MSG THRU P210Ø-BUILD-MSG-EXIT
  ELSE MOVE CURR-OBS TO LAST-OBS(I)
  MOVE ZERO TO TOTAL-ELAPSED(I) LONG-RUNNING-FLAG
  PERFORM P220Ø-OK THRU P220Ø-OK-EXIT
END-IF
END-EVALUATE.
MOVE SPACES TO TS-TABLE.
MOVE RESOURCE-CHECKS(I) TO TS-RES-NAME.
MOVE MQ-LIMIT(I) TO TS-DESIRED
MOVE LAST-OBS(I) TO TS-LAST-OBS.
MOVE TOTAL-ELAPSED(I) TO TS-TOTAL-ELAPSED.
MOVE SINCE-CHECKED(I) TO TS-SINCE-CHECKED.
MOVE STATUS-FLAG(I) TO TS-STATUS-FLAG.
COMPUTE H = I + 1.
EXEC CICS WRITEQ TS QUEUE('S5ØPRESC')
  FROM(TS-TABLE) ITEM(H) REWRITE
END-EXEC.
GO TO P2ØØØ-PROC-EXIT.
P201Ø-CONN.
  EXEC CICS INQUIRE CONNECTION(RES-NAME(I))
     CONNSTATUS(CURR-OBS)
END-EXEC.
*  IF CURR-OBS = DFHVALUE(ACQUIRED)
*     MOVE DFHVALUE(AVAILABLE) TO LAST-OBS(I)
*  ELSE IF CURR-OBS = DFHVALUE(AVAILABLE)
*     MOVE DFHVALUE(ACQUIRED) TO LAST-OBS(I)
*  END-IF.
P201Ø-CONN-EXIT.
EXIT.
P201Ø-TRAN.
```

MOVE ZERO TO LONG-RUNNING-FLAG.
EXEC CICS COLLECT STATISTICS TRANSACTION(RES-NAME(I))
   SET(ADDRESS OF DFHXMRDS)
END-EXEC.
MOVE XMRAC TO CURR-OBS.
P2010-TRAN-CONT.
   IF RES-NAME-4(I) = 'I91H' OR
   RES-NAME-4(I) = 'I91J' OR
   RES-NAME-4(I) = 'I91K' OR
   RES-NAME-4(I) = 'I91Y' OR
   RES-NAME-4(I) = 'I9MR'
      PERFORM P5000-INQUIRE THRU P5000-INQUIRE-EXIT.
P2010-TRAN-EXIT.
EXIT.
P2010-TX.
   MOVE ZERO TO LONG-RUNNING-FLAG.
   EXEC CICS COLLECT STATISTICS TRANSACTION(RES-NAME(I))
      SET(ADDRESS OF DFHXMRDS)
END-EXEC.
   MOVE XMRAC TO CURR-OBS.
P2010-TX-CONT.
   PERFORM P5000-INQUIRE THRU P5000-INQUIRE-EXIT.
P2010-TX-EXIT.
EXIT.
P2010-PU.
   EXEC CICS HANDLE CONDITION NOTFND(P2010-PU-COLLECT-UP)
END-EXEC.
   MOVE RES-NAME(I) TO PU-ID.
   MOVE ZERO TO PU-NO.
   PERFORM P2010-PU-COLLECT THRU P2010-PU-COLLECT-EXIT
      100 TIMES.
   GO TO P2010-PU-EXIT.
P2010-PU-COLLECT.
   EXEC CICS COLLECT STATISTICS TERMINAL(TERM-ID)
      SET(ADDRESS OF DFHAØ6DS)
END-EXEC.
   MOVE AØ6TEOT TO NUM-TRAN-X.
   ADD NUM-TRAN-N TO CURR-OBS.
P2010-PU-COLLECT-UP.
   ADD 1 TO PU-NO.
P2010-PU-COLLECT-EXIT.
EXIT.
P2010-PU-EXIT.
EXIT.
Ø8564Ø P2010-MQS.
   MOVE MQ-LIMIT(I) TO WS-MQ-DEPTH-X.
   MOVE 'MQ' TO KEY24Ø2-ZIHUY-MASHAV.
   MOVE RES-NAME(I) TO KEY24Ø2-SHEM-LOGI.
   MOVE KEY24Ø2-KEY TO TAB24Ø2-KEY.
   PERFORM P4000-READ-DPT.
IF TAB2402-REPLY = ZERO
   NEXT SENTENCE
ELSE GO TO P2000-PROC-EXIT.

MOVE TAB2402-QUEUE-NAME TO MQOD-OBJECTNAME
MOVE MQMD-REPLYTOQMGR TO MQOD-OBJECTQMGRNAME.
MOVE MQOO-INQUIRE TO W03-OPTIONS.
CALL 'MQOPEN' USING W03-HCONN,
MQM-OBJECT-DESCRIPTOR,
W03-OPTIONS,
W03-HOBJ,
W03-COMPCODE,
W03-REASON.

IF W03-COMPCODE NOT = MQCC-OK
   IF W03-COMPCODE = MQRC-UNKNOWN-OBJECT-NAME
      GO TO P2000-PROC-EXIT
   END-IF
   MOVE W03-COMPCODE TO TEXT4-COMP
   MOVE W03-REASON TO TEXT4-REASON
   MOVE TEXT4 TO LOGTEXT
   EXEC CICS WRITEQ TD QUEUE('DJNS') FROM(LOGHEADR)
   LENGTH(TD-BUFFLEN)
   END-EXEC
   MOVE MQIA-Q-TYPE TO SELECTORS(1).
   MOVE MQIA-CURRENT-Q-DEPTH TO SELECTORS(2).
   MOVE MQCA-Q-DESC TO SELECTORS(3).
   CALL 'MQINQ' USING W03-HCONN,
   W03-HOBJ,
   SELECTORCOUNT,
   SELECTORS-TABLE,
   INTATTRCOUNT,
   INTATTRS-TABLE,
   CHARATTLENGTH,
   CHARATTRS,
   W03-COMPCODE,
   W03-REASON.
   IF W03-COMPCODE NOT = MQCC-OK
MOVE WØ3-COMPCODE TO TEXT5-COMP
MOVE WØ3-REASON TO TEXT5-REASON
MOVE TEXT5 TO LOGTEXT
MOVE LENGTH OF LOGHEADR TO TD-BUFFLEN
EXEC CICS WRITEQ TD QUEUE('DJNS') FROM(LOGHEADR)
   LENGTH(TD-BUFFLEN)
END-EXEC
MOVE TAB24Ø2-QUEUE-NAME TO TEXT4A-Q-NAME
MOVE TEXT4A TO LOGTEXT
MOVE LENGTH OF LOGHEADR TO TD-BUFFLEN
EXEC CICS WRITEQ TD QUEUE('DJNS') FROM(LOGHEADR)
   LENGTH(TD-BUFFLEN)
END-EXEC
IF WØ3-COMPCODE = MQRC-SELECTOR-NOT-FOR-TYPE
  GO TO P2Ø1Ø-MQ-CLOSE
END-IF
GO TO P2Ø1Ø-MQ-CLOSE
END-IF.
MOVE INTATTRS(2) TO CURR-OBS.
P2Ø1Ø-MQ-CLOSE.
CALL 'MQCLOSE' USING WØ3-HCONN
  WØ3-HOBJ
  MQCO-NONE
  WØ3-COMPCODE
  WØ3-REASON.
IF WØ3-COMPCODE NOT = MQCC-OK
  MOVE WØ3-COMPCODE TO TEXT6-COMP
  MOVE WØ3-REASON TO TEXT6-REASON
  MOVE TEXT6 TO LOGTEXT
  MOVE LENGTH OF LOGHEADR TO TD-BUFFLEN
  EXEC CICS WRITEQ TD QUEUE('DJNS') FROM(LOGHEADR)
   LENGTH(TD-BUFFLEN)
END-EXEC
MOVE TAB24Ø2-QUEUE-NAME TO TEXT4A-Q-NAME
MOVE TEXT4A TO LOGTEXT
MOVE LENGTH OF LOGHEADR TO TD-BUFFLEN
EXEC CICS WRITEQ TD QUEUE('DJNS') FROM(LOGHEADR)
   LENGTH(TD-BUFFLEN)
END-EXEC
END-IF.
IF CURR-OBS >= WS-MQ-DEPT9
  PERFORM P5ØØØ-INQUIRE THRU P5ØØØ-INQUIRE-EXIT.
Ø894ØØ P2Ø1Ø-MQS-EXIT.
Ø895ØØ EXIT.
Ø894ØØ P2Ø1Ø-FILE.
EXEC CICS HANDLE CONDITION FILENOTFOUND(P2Ø1Ø-FILE-EXIT)
END-EXEC.
EXEC CICS INQUIRE FILE(RES-NAME(1)) OPENSTATUS(CURR-OBS)
END-EXEC.
Ø894ØØ P2Ø1Ø-FILE-EXIT.
P2100·BUILD·MSG.
MOVE ALT·STAT TO LOG·STAT
IF STATUS·FLAG(I) = STATUS·CLOSED
MOVE STATUS·OPEN TO STATUS·FLAG(I)
MOVE 'IS INACTIVE' TO MQ·MSG
MOVE 'OPEN ' TO MQ·EVENT
ELSE
MOVE STATUS·STILL·OPEN TO STATUS·FLAG(I)
MOVE 'STILL INACTIVE' TO MQ·MSG LOG·STAT
MOVE 'STILL' TO MQ·EVENT
END-IF
ADD VALID·INTERVAL TO TOTAL·ELAPSED(I)
PERFORM P3000·SEND·MQ THRU
P3000·SEND·MQ·EXIT.
P2100·BUILD·MSG·EXIT.
EXIT.
P2200·OK.
IF STATUS·FLAG(I) = STATUS·OPEN OR STATUS·STILL·OPEN
MOVE STATUS·CLOSED TO STATUS·FLAG(I)
MOVE 'NOW ACTIVE' TO MQ·MSG LOG·STAT
MOVE 'CLOSE' TO MQ·EVENT
PERFORM P3000·SEND·MQ THRU
P3000·SEND·MQ·EXIT
END-IF.
P2200·OK·EXIT.
EXIT.
P2500·DELAY.
IF SHOW·HR = 23
IF SHOW·MIN >= 58
PERFORM P2700·END·OF·DAY THRU P2700·END·OF·DAY·EXIT
VARYING I FROM 1 BY 1
UNTIL I > NUM·ACTIVE
MOVE 1 TO FIRST·TIME·FLAG
EXEC CICS RETURN
END·EXEC
END-IF.
EXEC CICS DELAY FOR MINUTES(DAY·TIME)
END·EXEC.
P2500·DELAY·EXIT.
EXIT.
P2700·END·OF·DAY.
IF RES·NAME(I) = 'TIMESKED'
OR RES·NAME·STAR(I) = '*'
GO TO P2700·END·OF·DAY·EXIT
END-IF.
IF STATUS·FLAG(I) = STATUS·OPEN OR STATUS·STILL·OPEN
MOVE STATUS-CLOSED TO STATUS-FLAG(I)
MOVE 'END OF DAY' TO MQ-MSG LOG-STAT
MOVE 'CLOSE' TO MQ-EVENT
PERFORM P3000-SEND-MQ THRU
    P3000-SEND-MQ-EXIT
END-IF.
P2700-END-OF-DAY-EXIT.
EXIT.
P3000-SEND-MQ.
IF RES-TYPE(I) = MQS
    MOVE CURR-OBS TO MQ-DEPTH
    MOVE 'Q DEPTH = ' TO MQ-DEPTH-CONST
ELSE MOVE SPACES TO MQ-MQS-ONLY.
MOVE SHOWDATE TO MQ-DATE.
MOVE SHOWTIME TO MQ-TIME.
MOVE SYS-ID TO MQ-SYSID.
MOVE RES-NAME(I) TO MQ-RES-NAME LOG-NAME.
MOVE RES-TYPE(I) TO MQ-RES LOG-TYPE.
MOVE TOTAL-ELAPSED(I) TO LOG-TIME.
MOVE SAVEQ TO MQMD-REPLYTOQ.
MOVE MQMD-REPLYTOQ TO MQOD-OBJECTNAME.
MOVE MQMD-REPLYTOQSMGR TO MQOD-OBJECTQMGRNAME.
MOVE MQMT-REPLY TO MQMD-MSGTYPE.
MOVE MQFMT-STRING TO MQMD-FORMAT.
MOVE SPACES TO MQMD-REPLYTOQSMGR.
MOVE LOW-VALUES TO MQMD-MSGID.
*    COMPUTE MQPMO-OPTIONS = MQPMO-SYNPONOT +
    MQPMO-PASS-IDENTITY-CONTEXT.
*    MOVE WØ3-HOBJ-CHECKQ TO MQPMO-CONTEXT.
MOVE LENGTH OF VANTIVE-MSG TO WØ3-BUFFLEN.
CALL 'MQPUT1' USING WØ3-HCONN
MQOD
MQMD
MQPMO
WØ3-BUFFLEN
VANTIVE-MSG
WØ3-COMPCODE
WØ3-REASON.
MOVE TEXT2 TO LOGTEXT.
MOVE LENGTH OF LOGHEADR TO TD-BUFFLEN.
EXEC CICS WRITEQ TD QUEUE('DJNS') FROM(LOGHEADR)
    LENGTH(TD-BUFFLEN)
END-EXEC.
IF WØ3-COMPCODE NOT = MQCC-OK
    MOVE WØ3-COMPCODE-X TO TEXT3-COMP
    MOVE WØ3-REASON-X TO TEXT3-REASON
    MOVE TEXT3 TO LOGTEXT.
    MOVE LENGTH OF LOGHEADR TO TD-BUFFLEN
    EXEC CICS WRITEQ TD QUEUE('DJNS') FROM(LOGHEADR)
    LENGTH(TD-BUFFLEN)
END-IF.
IF RES-TYPE(I) = MQS
  IF WORK-TRAN-ID = MQ-TRAN(I)
    GO TO P5000-LONG-RUNNING
  END-IF
END-IF.
ADD LENGTH OF TRAN-ID TO TASK-LIST-POINTER-REDEF.
P5010-INQUIRE-EXIT.
EXIT.
P8000-INIT.
EXEC CICS ASSIGN SYSID(SYS-ID)
END-EXEC.
EVALUATE SYS-ID
WHEN TEST-SYS
  MOVE 'MQST' TO MQMD-REPLYTOQMGR
  MOVE 'VAN.EVENT_LOG_SYS1' TO MQMD-REPLYTOQ
WHEN Z-SYS
  MOVE 'MQST' TO MQMD-REPLYTOQMGR
  MOVE 'VAN.EVENT_LOG_SYS1' TO MQMD-REPLYTOQ
WHEN QA-SYS
  MOVE 'MQSV' TO MQMD-REPLYTOQMGR
  MOVE 'VAN.EVENT_LOG_SYS1' TO MQMD-REPLYTOQ
WHEN SYST-SYS
  MOVE 'MQST' TO MQMD-REPLYTOQMGR
  MOVE 'VAN.EVENT_LOG_SYS1' TO MQMD-REPLYTOQ
WHEN TS13-SYS
  MOVE 'MQST' TO MQMD-REPLYTOQMGR
  MOVE 'VAN.EVENT_LOG_SYS1' TO MQMD-REPLYTOQ
WHEN PROD-SYS
  MOVE 'MQSC' TO MQMD-REPLYTOQMGR
  MOVE 'VAN.EVENT_LOG_SYS3' TO MQMD-REPLYTOQ
END-EVALUATE.
MOVE MQMD-REPLYTOQ TO SAVEQ.
EXEC CICS IGNORE CONDITION QIDERR
END-EXEC.
EXEC CICS DELETEQ TS QUEUE('S50PRESC')
END-EXEC.
EXEC CICS HANDLE CONDITION QZERO(P8000-FIN)
END-EXEC.
MOVE 1 TO I.
MOVE DFHVALUE(OPEN) TO STAT-1.
EXEC CICS SET TDQUEUE('RESC') OPENSTATUS(STAT-1)
END-EXEC.
EXEC CICS HANDLE CONDITION QZERO(P8050-FIN)
END-EXEC.
MOVE ZERO TO I.
EXEC CICS WRITEQ TS QUEUE('S50PRESC')
FROM(TABLE-HEADER)
END-EXEC.
PERFORM P8050-READ THRU P8050-READ-EXIT 800 TIMES.
MOVE DFHVALUE(CLOSED) TO STAT-1.
EXEC CICS SET TDQUEUE('RESC') OPENSTATUS(STAT-1) END-EXEC.
MOVE HIGH-VALUES TO RESOURCE-TABLE(I).
MOVE Text1 TO LOGTEXT
MOVE LENGTH OF LOGHEADER TO TD-BUFFLEN
EXEC CICS WRITEQ TD QUEUE('DJNS') FROM(LOGHEADER) LENGTH(TD-BUFFLEN) END-EXEC.

P8ØØØ-INIT-EXIT.
EXIT.

P8Ø5Ø-READ.
EXEC CICS READQ TD QUEUE('RESC') INTO(RESOURCE-RECORD) END-EXEC.
IF REC-TYPE = '*' GO TO P8Ø5Ø-READ.
IF REC-TYPE = NA
   ADD 1 TO I
   MOVE INPUT-DATA TO RESOURCE-CHECKS(I)
   MOVE I TO NUM-ACTIVE
   MOVE ZERO TO TOTAL-ELAPSED(I) LAST-OBS(I) SINCE-CHECKED(I)
   MOVE STATUS-CLOSED TO STATUS-FLAG(I)
   MOVE SPACES TO TS-TABLE
   MOVE RESOURCE-CHECKS(I) TO TS-RES-NAME
   MOVE MQ-LIMIT(I) TO TS-DESIRED
   MOVE LAST-OBS(I) TO TS-LAST-OBS
   MOVE TOTAL-ELAPSED(I) TO TS-TOTAL-ELAPSED
   MOVE SINCE-CHECKED(I) TO TS-SINCE-CHECKED
   MOVE STATUS-FLAG(I) TO TS-STATUS-FLAG
*P8Ø5Ø-WRITE-Q.
EXEC CICS WRITEQ TS QUEUE('S5ØPRESC') FROM(TS-TABLE) END-EXEC
ELSE IF REC-TYPE = DA
   IF REC-TITLE = HOLIDAY MOVE INPUT-DATA TO RESOURCE-TIMES(I, 8) GO TO P8Ø5Ø-READ
   ELSE MOVE 1 TO Q GO TO P8Ø5Ø-LOOP END-IF
END-IF
END-IF.
P8Ø5Ø-LOOP.
IF RES-DAY-NUM(Q) > 0 AND RES-DAY-NUM(Q) < 9
   MOVE RES-DAY-NUM(Q) TO X
   MOVE INPUT-DATA TO RESOURCE-TIMES(I, X) EVALUATE X WHEN 1
   MOVE SUNDAY TO DAY-NAME(I, X)
CICSPlex SM dynamic workload management – workloads

In this article we will look at the various types of workload that can be balanced using dynamic workload management. CICS, along with CICSPlex SM, provides the ability to route many types of CICS request. In the following sections we will discuss each of these in turn.

In the Figures we have two layers of regions: TORs and AORs. The lines connecting each region correspond to CICS connections. Workload management criteria have been defined via WLMDEF, TRANGRP, WLMGROUP, WLMSPEC, and associated system groups. That information is illustrated in the
top left-hand corner of the diagrams. The RDO attributes that control dynamic routing are identified to the right of the diagram.

**DYNAMIC TRANSACTION ROUTING**
- Available since: CICS 3.3
- Routing model: DTRPGM
- SystemGroup: AORSET
- SystemGroup: TORSET.

```
trangrp(x) = {tranid}
(userid,luname,trangrp,processtype)->SystemGroup
Associated with SystemGroup'
```

Requestors are out in the network, TORs are routers, and AORs are targets.

In Figure 1 we see an invocation of a transaction by an end user. Since the transaction is defined as dynamic (or not defined and controlled by the SIT parm), routing is invoked. CICS passes tranid, userid, and luname to the routing exit. First the associated
trangroup is identified (specific or default). If an affinity exists, the target is returned to CICS. Next, via the separation criteria, the target AORs are identified. Weights are then calculated and the region with the lowest weight chosen. If an affinity is defined but not active, an affinity element is created. The load count for the specific region is incremented. The target is then returned to CICS, which routes the request to the target region.

On completion of the transaction in the AOR, control is passed back to the AOR and routing is invoked again (terminate or abend termination). The load count for the region is decremented, and any abend is noted for use by abend avoidance.

EXEC CICS START TERMID

- Available since: CICS TS 1.3
- Routing model: DTRPGM
- SystemGroup: AORSET

```
trongrp(x) = {tranid}
(userid,luname,trangrp,processtype)->SystemGroup
Associated with SystemGroup'
```

EXEC CICS START
TRANID(abcd)
TERMID(EIBTRMID)

*Figure 2: EXEC CICS START TERMID*
• SystemGroup: TORSET.
AORs are requestors, TORs are routers, and AORs are targets.

The ability to route START TERMID requests was provided in CICS TS 1.3. Prior to that, state data was maintained by CICS in the requesting region. Consequently, the work had to be routed back to the same AOR. In CICS TS 1.3, the state data is shipped to the TOR and dynamic routing can occur. This therefore requires that all CICS regions in Figure 2 are at CICS TS 1.3 level.

EXEC CICS START:
• Available since: CICS TS1.3
• Routing model: DSTRPGM
• SystemGroup: AORSET
• SystemGroup: AORSET.
AORs are requestors, routers, and targets.

trangrp(x) = {tranid}
(userid,luname,trangrp,processtype)->SystemGroup
Associated with SystemGroup

Figure 3: EXEC CICS START
Dynamic routing of EXEC CICS START requests was introduced in CICS TS 1.3. Here we have an application requesting a START. If we assume that the AORs are clones of each other, then every region is a potential requestor, router, and target! All regions must be at least CICS TS 1.3 level.

In Figure 3, we have a requestor who requests execution of an asynchronous request. After the request, he has no idea of when that request is processed because CICS could dispatch the START later and no termination flow comes back to the requestor. The load count is incremented just prior to dispatch in the target (rtinit) and decremented at the target on termination (rtterm/rtabd).

There is also the concept of originating region, ie one cannot daisychain requests. Once a routing decision has been made to a target, that target cannot invoke dynamic routing yet again.

CICS BUSINESS TRANSACTION SERVICES REQUESTS:
• Available since: CICS TS 1.3

\[
\text{trangrp}(x) = \{\text{tranid}\} \\
(\text{userid},\text{luname},\text{trangrp},\text{processtype}) \rightarrow \text{SystemGroup} \\
\text{Associated with SystemGroup’}
\]

\[
\text{EXEC CICS RUN} \\
\text{ACTIVITY(name)} \\
\text{<TRANID(abcd)>} \\
\text{ASYNCH}
\]

\[\begin{align*}
\text{define tranid(abcd)} \\
\text{dynamic(yes)} \\
\text{routable(yes)}
\end{align*}\]

Figure 4: CICS Business Transaction Services Requests
• Routing model: DSTRPGM
• SystemGroup: AORSET
• SystemGroup: AORSET.

AORS are routers, requestors, and targets.

In Figure 4, CBTS RUN ASYNCH requests result in calls to routing and a target is chosen. This may also occur when an activity is paged back into CICS following the triggering of an event.

DYNAMIC PROGRAM LINK
• Available since: CICS TS1.3
• Routing model: DTRPGM
• SystemGroup: AORSET
• SystemGroup: TORSET.

\[
\text{trangrp}(x) = \{\text{tranid}\} \\
(\text{userid}, \text{luname}, \text{trangrp}, \text{processtype}) \rightarrow \text{SystemGroup} \\
\text{Associated with SystemGroup}'
\]

Web, EXCI, ECI specifying prog(xyz)
<tranid(abcd)>

\[
\begin{align*}
\text{define program}(xyz) \\
<\text{tranid}(abcd)> \\
\text{dynamic}(yes)
\end{align*}
\]

\[
\begin{align*}
\text{define program}(xyz) \\
\text{dynamic}(no)
\end{align*}
\]

\[
\begin{align*}
\text{define tranid}(abcd) \\
\text{program(dfhmirrs)}
\end{align*}
\]

Figure 5: Dynamic Program Link
Requestors are out in the network, TORs are routers, and AORs are targets.

The ability to dynamically route DPL requests was introduced in CICS TS 1.3. However, unlike the other types, only the router/requester is affected. The targets can be older systems (eg CICS 4.1). Since this is a way for Web traffic entering into CICS, and Web is so unpredictable in volume, the introduction of a CICS TS1.3 TOR is all that’s required to dynamically balance this type of workload.

Note that CICSPlex SM defines routing criteria with respect to transaction ids, not program names. CICS associates tranids with the program by the following priority of tranid on request; tranid on program definition; CSMI. Note that if the tranid is not CSMI, then there must be a transaction definition for that tranid resolving to the mirror program in the target regions. This is illustrated in Figure 5.

```
trangrp(x) = {tranid}
  (userid,luname,trangrp,processstype)->SystemGroup
Associated with SystemGroup'
```

```
EXEC CICS LINK
PROGRAM(xyz)
  <TRANID(abcd)>
```

**Figure 6: Peer-to-peer dynamic DPL**
PEER-TO-PEER DYNAMIC DPL

- Available since: CICS TS1.3
- Routing model: DTRPDM
- SystemGroup: AORSET
- SystemGroup: AORSET.

AORs are requestors, routers, and targets.

Although Figure 6 is topologically equivalent to earlier examples, there are some differences. Routing occurs as above. The differences are in goal management.

Goal management is by tranid (both within CICSPlex SM and MVS WLM). Mapping of program to tranid is performed by CICS. Prior to the invocation of routing, CICS classifies the tranid via a call to the MVS WLM. For simple inbound DPLs, it is reasonable to assume that the goal for this tranid is the goal for the program (ie one in one out). A Performance Index can therefore be meaningfully calculated.

In the peer-to-peer case, one cannot make such an assumption, since the classification is of the invoking transaction (ie the requestor), not the invoked program. Clearly the requestor could invoke the program many times during one invocation of the requesting transaction, therefore for peer-to-peer, performance indexes are not calculated. Inbound or peer-to-peer is determined by inspection of EIBTRNID. If this is the routing tranid, then the request is inbound.

There is also the concept of originating region, ie one cannot daisychain requests. Once a routing decision has been made to a target, that target cannot invoke dynamic routing yet again.

DYNAMIC WORKLOAD MANAGEMENT OF ENTERPRISE JAVA BEANS

- Available since: CICS TS2.1
- Routing model: DSRTPGM
• SystemGroup: AORSET
• SystemGroup: TORSET.

Requestors are out in the network, TORs are routers, and AORs are targets.

In CICS TS 2.1 the ability to execute Enterprise Java Beans in a logical CICS server was introduced. EJBs are installed into AORs and the generic listener TCP/IP address published in the JNDI namespace. Client code can therefore look up the EJB home and be session managed via a DNS to a specific CICS listener region. Dynamic routing of the request to the chosen AOR occurs. This mechanism is somewhat analogous to VTAM generic resource balancing and DSRTPGM dynamic routing. Identification of the target is a two-stage process, as we shall see.
In the listener region a tranid is associated with the bean method via information contained in a Requestmodel definition. This provides the tranid used for making the routing decision. On arrival at the target region, since there may be multiple Corbaservers, the specific CorbaServer is identified again with reference to the requestmodel definition. This is illustrated in Figure 8.

Only the initial invocation is a request to routing to balance (rtsel). If state is maintained in the AOR, then CICS maintains this relationship and subsequent routing calls are made to simply tell routing the load implication (rtntfy).

Figure 8: Listener region
DYNAMIC BRIDGE REQUESTS:

- Available since: CICS TS2.2
- Routing model: DTRPGM
- SystemGroup: AORSET
- SystemGroup: TORSET.

Requestors are out in the network, TORs are routers, and AORs are targets.

CICS TS 2.2 introduced the ability to route dynamic bridge requests. Workload separation is supported via trangroup and

Figure 9: Dynamic Bridge requests
**Figure 10: Dynamic Bridge requests**

![Diagram](image)

**Figure 11: CICS Releases summary**

<table>
<thead>
<tr>
<th>Workload type</th>
<th>Router</th>
<th>Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTR</td>
<td>3.3</td>
<td>4.1</td>
</tr>
<tr>
<td>DPL</td>
<td>5.3</td>
<td>4.1</td>
</tr>
<tr>
<td>START</td>
<td>5.3</td>
<td>5.3</td>
</tr>
<tr>
<td>CBTS</td>
<td>5.3</td>
<td>5.3</td>
</tr>
<tr>
<td>EJB</td>
<td>6.1</td>
<td>6.1</td>
</tr>
<tr>
<td>Bridge</td>
<td>6.2</td>
<td>6.1</td>
</tr>
</tbody>
</table>
userid, but Luname is not predictable unless the customer site codes an exit that makes it so. BRIDGE affinities managed by CICS code CICSPlex SM are initially called for route select. Thereafter only for route notify. See Figures 9 and 10.

ROUTER AND TARGET RELEASES

In all the above we have identified the minimum release that a CICS system needs to be to support the routing function. What happens if this is not satisfied? As part of the potential target determination, CICSPlex SM checks the target systems release level. If that release level is not appropriate, then it is removed from the potential target list. A summary of CICS Releases is provided in Figure 11.

NEXT ARTICLE

In the next article we will look at implementing dynamic workload management in a running CICS environment.

Dr Paul Johnson
CICS Transaction Server Systems Management Planning/Development
IBM (UK) © IBM 2003

Book review – Murach’s CICS Desk Reference

Murach’s CICS Desk Reference, written by Raul Menendez and Doug Lowe, is published by Mike Murach and Associates. The 592-page book is aimed squarely at experienced programmers who write or maintain CICS programs. It focuses on CICS TS 1.3 and 2.2, because IBM has dropped (or is dropping) support for earlier versions.

The book is divided into four main sections. The first provides CICS programming guidelines and has chapters on program design, programming fundamentals, JCL, testing and debugging, and model programs.
The second section is a CICS command reference, looking at the syntax of commands and then over 270 pages looking at the actual commands themselves. This includes the code itself, the syntax, and a description of the options. There's also exceptional conditions, notes and tips, and coding examples.

The third section looks at Basic Mapping Support (BMS), focusing on definitions for 3270 displays and creating HTML documents from BMS maps.

The third section, some 50 pages, comprises useful AMS commands, CICS resource definition, service transactions (CEMT commands, etc), and a handy reference table.

All in all, a very useful reference book.

© Xephon 2003

CICS questions and answers

Q  Is there a way to get the Jobname CICS is running under?
A  EXEC CICS INQUIRE SYSTEM JOBNAME(jobname).

Q  I operate SAP R/2 at CICS/MVS ESA 4.1. I want to prevent a single user logging on to CICS with the same user ID several times. How can I do this?
A  The SIT parameter SNSCOPE allows you to prevent the same user multi-logging on to CICS. Setting SNSCOPE=CICS will only allow each userid to sign on once in each CICS region.

Q  What's the difference between CICS Web Support and CICS Web Support with WebServer plug-in?
A  CICS Web Support uses CICS as a Web server, accepting HTTP requests from and sending HTTP responses to Web clients through a TCPIPSERVICE. This uses a URM as an analyser, and the Web task runs under a Web alias
transaction. CICS Web Support with WebServer plug-in uses the plug-in instead of the analyser. This plug-in runs in the HTTP Server for OS/390, then uses EXCI to access the CICS Business Logic Interface. This will run under a mirror transaction. CICS Web Support connects directly into CICS, allowing the data input and output to exceed the 32KB limit by using the Web API. However, CWS with the WebServer plug-in, because it uses EXCI with a COMMREA, still has the 32KB limitation. The direct connection uses the analyser within CICS to determine the format of the request, whereas the plug-in uses directives within the configuration file of the HTTP Server. Essentially, the first turns CICS into a WebServer, whereas the second opens CICS up to EXCI requests from the WebServer.

If you have any CICS-related questions, please send them in and we will do our best to find answers. Alternatively, e-mail them directly to cicsq@xephon.net.

© Xephon 2003
NEON Systems has announced that its Shadow JDBC Adapter for mainframe integration has passed the WebSphere Self-Testing process and will be added to IBM’s Self-Tested Software support page. The IBM-sponsored programme facilitates self-testing of WebSphere complementary software through an IBM-endorsed testing process.

Shadow software can be deployed with WebSphere to provide JCA or JDBC access to mainframe data sources and transaction environments, supporting DB2, CICS/TS, IMS/TM, IMS/DB, V$SAM, ADABAS, Natural/ACI, flat files, IDMS, and other z/OS mainframe data and transactional sources.

For further information contact:
NEON Systems, 14100 Southwest Freeway, Suite 500, Sugarland, TX 77478, USA.
Tel: (281) 491 4200.

---

As an alternative platform for running CICS, UMX Technologies has announced Mainframe in a Box, a small to medium-sized mainframe running on a specially designed Intel-based UMX Server using Microsoft Windows 2000 or XP as the graphical user interface.

The installed software mainframe is UMX Virtual Mainframe V4.2 microcode engine, which functions between the IBM operating system and the common Intel-based hardware to ‘virtualize’ the hardware to the software.

Mainframe in a Box uses the original IBM operating system and existing applications, without a single modification. All operating systems (OS/390, z/OS, VM, z/VM, and VSE) and CICS, PL/I, IMS, COBOL, and DB2 applications run on this new mainframe.

PCI add-in cards support ESCON and Parallel Channel extension technologies to provide mainframe connectivity to legacy devices and other mainframes.

For further information contact:
UMX Technologies, Kruislaan 400 NL-1098 SM, Amsterdam, The Netherlands.
Tel: (+31)20 888 4044.
URL: http://www.umxtech.com/index0.html.

---

IBM has released Tivoli System Automation for OS/390 (SA OS/390) under the Tivoli Environment-Managed Licensing Model, which means pricing and licensing are based on what is managed rather than how the software is implemented.

The software is designed to automate I/O, processor, and system operations and includes canned automation for CICS, IMS, IBM Tivoli Workload Scheduler, and DB2.

Key functions include Parallel Sysplex application automation, policy-based self-healing, integration, processor operations (ProcOps) and I/O operations, and SAP R/3 high-availability automation.

For further information contact your local IBM representative.

---