# 224

# CICS update

*July 2004*

## In this issue

## Published by

Xephon Inc
PO Box 550547
Dallas, Texas 75355
USA

Phone: 214-340-5690
Fax: 214-341-7081

## Editor

Trevor Eddolls
E-mail: trevore@xephon.com

## Publisher

Nicole Thomas
E-mail: nicole@xephon.com

## Subscriptions and back-issues

A year's subscription to *CICS Update*, comprising twelve monthly issues, costs $270.00 in the USA and Canada; £175.00 in the UK; £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 2000 issue, are available separately to subscribers for $24.00 (£16.00) 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 $160 (£100 outside North America) per 1000 words and $80 (£50) per 100 lines of code for the first 200 lines of original material. The remaining code is paid for at the rate of $32 (£20) per 100 lines. 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.



*Printed in England.*

# The Kill function

CICS has recently been enhanced to provide support for new System Programming Interface (SPI) functionality – the Kill function. This article describes the background to the implementation of support for Kill within CICS, the way in which the function has been added to CICS, and the means by which Kill may be used in the CICS environment.

## BACKGROUND TO CICS TASK MANAGEMENT BEFORE KILL

CICS provides a number of ways to manage tasks within the system. For example, transaction definitions can include values for DTIMOUT. This definitional parameter specifies whether deadlock timeout is to be applied to the task. If the execution of the task gets suspended (for example through lack of storage), an automatic purge of the task is initiated should the task remain suspended for longer than the DTIMOUT value. In tandem with this, the transaction definition setting SPURGE specifies whether a transaction is system purgeable or not. SPURGE=NO prevents a task from being purged by the deadlock timeout (DTIMOUT) facility, an EXEC CICS ... PURGE command, TWAOCT (Cancel Task) being set in the node error program (NEP), or a CEMT SET ... PURGE command. SPURGE=YES allows such purges to go ahead as far as the user is concerned. CICS may, however, prevent the purge if it is not safe to allow a purge at the point the task has reached.

In addition to this automatic task management, the CICS System Programming Interface provides various mechanisms for the removal of tasks from a CICS system. Using the CEMT command, tasks may be purged or forcepurged from CICS. A purge is the recommended way to terminate a task as it is then removed from the system at an appropriate time, when CICS can guarantee both its system integrity and data integrity. This means that the action of purging a task leaves CICS control blocks in a consistent state, and that data associated with the task being purged is not

corrupted as a result. However, it is not always possible to purge a task from CICS. Certain operations within CICS may not be able to allow a purge to be honoured. This is because a purge at that time would jeopardize the integrity of data. In addition, it is possible for tasks to be defined with a transaction definition of SPURGE=NO, meaning that they are not eligible to be purged from the CICS system. As such, a more extreme means of task removal may be used – this is the forcepurge option. A forcepurge does protect CICS system integrity, but cannot guarantee that data integrity will be maintained if a task is forcepurged from the CICS system.

The recommendation has been (and indeed still is) that purge should be attempted first if a task or tasks are to be removed from CICS. Should a purge prove unsuccessful, and a subsequent forcepurge is attempted, unpredictable results may occur as a result.

In addition to the CEMT task, the SPI command EXEC CICS SET TASK PURGETYPE has been available for programmatical removal of tasks in a CICS system. This can specify CVDA values for purge and forcepurge. As with their use from CEMT, removing a task at the wrong time can result in a loss of data integrity or, in some circumstances, can cause the CICS system to abend. CICS will always defer purging until a task reaches a state where the system itself does not appear to be at risk. Unlike with forcepurge, purge will cause CICS also to wait until data integrity can be ensured.

An example of where purge and forcepurge requests are not allowed to proceed would be when CICS is waiting for a VSAM I/O request to complete as part of a File Control operation. If a task were purged while in this wait, the *Into* area used for the request could get reused or assigned to another task. When the VSAM I/O finally completed, the area would be overlaid as a result of the VSAM Get operation. This could cause system or data integrity problems. As such, CICS does not allow requests to purge or forcepurge tasks during this period of time.

It should be noted that SPURGE=NO does not prevent a task

from being purged by the read timeout (RTIMOUT) facility, an EXEC CICS SET TASK FORCEPURGE command, or a CEMT SET TASK(tranid) FORCEPURGE command. (This is also true for the equivalent EXEC CICS SET TERMINAL FORCEPURGE, EXEC CICS SET CONNECTION FORCEPURGE, CEMT SET TERMINAL FORCEPURGE, and CEMT SET CONNECTION FORCEPURGE commands.) SPURGE determines only the initial value, which can be changed by the transaction while it is running.

In addition to the options available from within CICS itself, the CICSPlex SM Web user interface and TSO end user interface allow the use of purge and forcepurge commands against task, terminal, and connection objects. CICSPlex SM also provides the equivalent programmatical means of purging or forcepurging tasks by means of the EXEC CPSM API.

## LOOPING TASKS

It is possible that a CICS task may enter a loop, which prevents normal processing from continuing within the CICS region. For example, a task could repeatedly execute instructions that fail to invoke CICS services, or else execute those CICS commands that do not cause CICS to reset the runaway timeout check for that task. If this is the case, CICS will no longer voluntarily receive control back from this looping task. Since CICS dispatching utilizes a cooperative processing model, a failure to invoke CICS dispatcher services like this will result in other tasks no longer being able to execute on the TCB that the looping program is running on. Typically, this will be under the quasi-reentrant (QR) TCB.

The QR TCB is used to subdispatch the majority of workload within a CICS region. It provides a serialized environment for non-threadsafe programs to execute in. It is possible, however, that a loop may occur within a program running under another CICS TCB. Now that CICS supports the Open Transaction Environment (OTE), open TCBs such as J8 and L8 TCBs are

also able to execute user programs within CICS, in parallel with those that are dispatched under the QR TCB. A loop on one of these TCBs would be serious for the program (and CPU) concerned, but it would be discrete from the rest of the CICS workload running in the system on the QR TCB and other open TCBs.

Normally, tasks trapped in such tight non-yielding loops will eventually be abended with an AICA abend code when CICS runaway timeout detection determines that the task has not returned control to CICS for a particular length of CPU time. This timeout period is either taken from the ICVR system parameter or it is overridden by the RUNAWAY parameter on the transaction definition, if this is specified. However, it is possible that the ICVR value (or RUNAWAY setting) may be set to a very high value (thus avoiding loop detection for a long period of time) or to 0 (which prevents runaway detection at all). It is also possible that a looping task may be in a yielding loop (where control is voluntarily returned to CICS, and the runaway timeout validation period is reset for the task). Although other CICS tasks may be able to run in the system, the yielding loop task's priority may prevent much other work from being executed. The looping task may also result in a shortage of CICS resources, thus preventing new work from being able to enter the systemor from being dispatched within CICS.

For the reasons outlined above, it can be seen that a looping task can bring normal processing within the affected CICS system to a halt. The implications of this are twofold. First, it might not be possible to identify the active task within CICS in order to be able to resolve the situation. For example, the looping task may prevent the QR TCB from dispatching CEMT to inquire on the tasks in the system. In other words, the act of listing the tasks within the CICS system may be unable to proceed because of the loop. Second, even if the rogue task (or tasks) is identifiable, it may not be possible to remove it from the system by means of purge or forcepurge commands because of the nature of the loop, the availability of the CICS dispatcher, and of the TCB under

which the looping task is executing. The only solution may be to cancel and restart the CICS system.

This is the situation that led to the development of the Kill function for use in task management within CICS.

## KILL IMPLEMENTATION WITHIN CICS

CICS has been enhanced to improve its ability to remove such rogue programs from a looping or stalled system. This change also minimizes the impact on the integrity of the CICS environment. It includes enhancements to the existing purge and forcepurge options, and support for a new type of task management option – the Kill function.

Support for the Kill function was added to CICS TS V2.2 by PTF UQ81836 (APAR PQ79277). Kill exploitation in turn utilizes enhancements made to CICS that were shipped by PTF UQ81795 (APARs PQ73474 and PQ79276).

Kill support has since been routed to CICS TS V2.3 by PTF UQ85743 (APAR PQ81514). The CICS enhancements were routed to CICS TS V2.3 by PTF UQ85695 (APAR PQ81760).

There are several aspects to this implementation. First, both the purge and forcepurge options have been enhanced significantly. As such, their use may now make it possible to remove an executing task from CICS, whereas it may not have done prior to these enhancements. In addition to the enhancements to purge and forcepurge support, should these options be unable to complete successfully, the new task management option of Kill may be used. Support for this has been added to the SET TASK, SET CONNECTION, and SET TERMINAL commands (for both CEMT and the EXEC CICS SPI, and for the EXEC CPSM API).

The Kill option does not provide any guarantee of CICS system or data integrity. It is intended as a last resort, when neither the purge nor the forcepurge commands have been able to remove a task from CICS. It should be noted that the Kill option can lead to unpredictable results, including the overwriting of data within

a CICS system or the abnormal termination of the CICS system itself. The type of work that can be performed as part of CICS recovery from a Kill operation is very much dependent on the environment(s) that are executed at the time of the Kill request.

When using the Kill option from CEMT, the EXEC CICS SPI, or the EXEC CPSM API, it is a requirement that at least a forcepurge must have been attempted first for the task in question. This is to ensure that less extreme options for task removal have been tried before Kill is attempted.

Using the Kill function via the CEMT command, or programmatically via the EXEC CICS SPI, requires that there is an environment within CICS that is available to execute the command. This means that the QR TCB must be available to dispatch the task issuing the Kill command. If the rogue task that is to be removed from CICS is looping under the QR TCB, it may well be that CICS cannot dispatch another task in order to allow Kill to be used. More fundamentally, it may not be possible to inquire on the looping task by means of CEMT or the SPI. This is because such an inquiry would itself require the QR TCB to be available under which to execute the commands. As such, there is the possibility that this kind of inquiry cannot be made directly from within the normal system programming environments under CICS or CICSPlex SM. To provide the ability of determining the active tasks in a CICS system (and hence deciding which is to be cancelled), a new transaction has been added to CICS. This is CEKL. It should be noted that CEKL is not a transaction in the true CICS sense of the word.

CEKL will be discussed in more detail in a future article.

## CONCLUSION

I hope this article has helped give a background to the implementation of Kill support within CICS.

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

# Helpful exit for shutdown assistant users – revisited

The article ‘Helpful exit for shutdown assistant users’ was published in *CICS Update*, issues 221 and 222, April and May 2004.

In the exit XXMATT (program CSXXMATT) the transaction class NLVTCL00 becomes assigned.

Figure 1 shows the definition of the NLVTCL00.

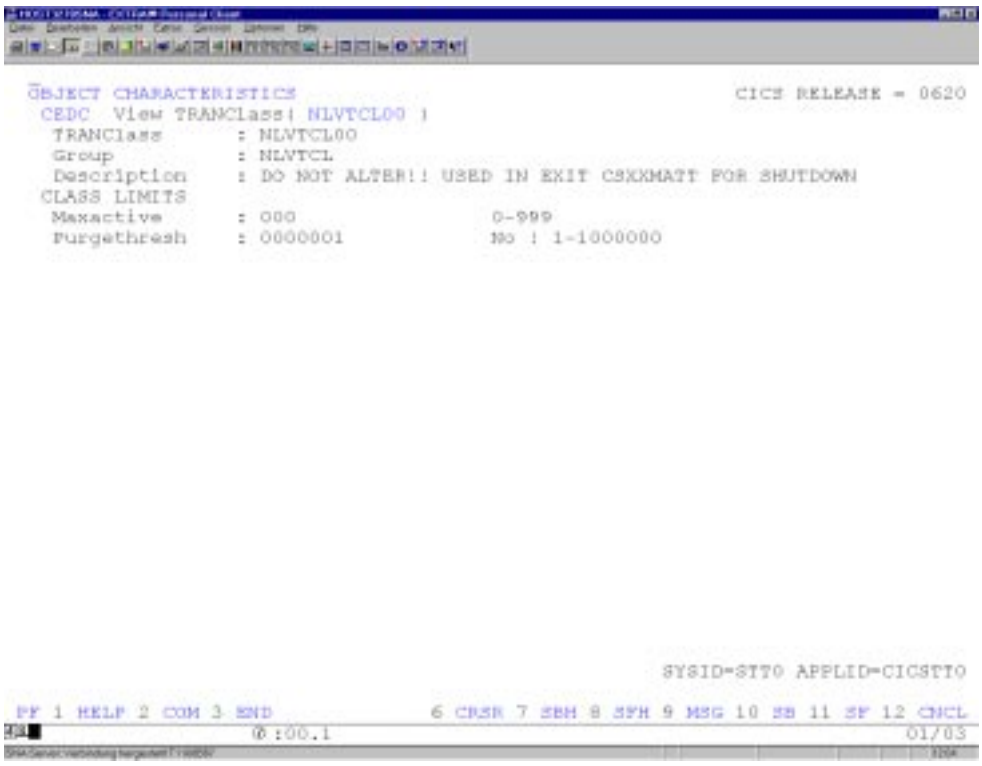


*Figure 1: Definition of the NLVTCL00*

A very important parameter is Maxactive = 000.

*Claus Reis*
*CICS Systems Programmer*
*Nuernberger Lebensversicherung AG (Germany)*

# Hints and tips for testing an APPC program

## PROBLEM ADDRESSED

At many CICS sites, it is quite common to encounter a program written in such a way that APPC commands issued often communicate with a partner program executing on a different platform.

In most of these cases, the CICS program is a server while the partner program is a client. The communication between these programs requires a defined APPC connection as well as a session on the host side. This might also involve other programs and tools, such as a communication program on the client side (beside the application program), which establishes the connection with an APPC session on the host, uses different network protocols (TCP/IP, SNA) etc, all of which might 'take the blame' if something goes wrong.

The application programmer's point of view on both sides in a session is concerned with data flow between the two – the client program sends the string of data and expects an answer from the server program, while the CICS server program receives the data and sends the answer back to the client.

Such a CICS-APPC transaction started by a remote end user creates a VTAM virtual terminal on which programs run. CICS programmers may find this to be quite inconvenient if testing, tracing or debugging is needed.

## PROPOSED SOLUTION

Pondering over this annoying situation, it was decided that keeping a record of the data received and sent back to the client program might be very useful. These data strings could be stored in several ways (VSAM file, DB2 table, etc) that need creating and maintaining in TSO.

In order to demonstrate the idea of monitored and controlled communication, a sample program was written (named PROGRAM1 in this example). Since the basic idea was to use mainly CICS 'resources', input and output data strings were to be stored in CICS temporary storage (TS).

The real strength of this approach is that it could be made use of even by standard CICS programs. When tracing and debugging the execution of a program, it was found to be very useful to write output to temporary storage at important (check) points in the logical flow of the program, along with the current values of some specified variables, and with the check point number. In cases where the program abends, it gives approximate (depending on the 'frequency' of the check points) information about the cause and the point in program where the abend occurred.

In order to view and handle such temporary storage, as well as test PROGRAM1, it was found to be quite helpful to write yet another auxiliary CICS terminal program with a map – PROGRAM2.

This auxiliary program shows each and every input and output string of PROGRAM1 by reading all of its input and output TS on the first entry in the program (or returning from PROGRAM1) and storing them in its own *all_strings_ts* temporary storage. In addition, PROGRAM2 has the ability to scroll through TS forward (PF8) and backward (PF7), delete the contents of TS (PF12), and delete only successfully ended transactions (I/O strings) (PF10). It can start PROGRAM1 by simply positioning the cursor at a field on the map next to the input string and pressing *Enter*.

The program is also capable of showing the entire contents of each input or output string on a blank screen, by simply positioning the cursor at the field next to it and pressing PF6. By pressing *Enter* afterwards, it starts the main program (PROGRAM1) with the possibly-altered input string – thus providing an easy way to test PROGRAM1.

## CODE

PROGRAM1 connected to a transaction TRN1 should have a structure similar to this one:

```
PROGRAM1: PROC(PTR1) OPTIONS(MAIN);
  -- /*****************************************/
  -- /** declaration of the program variables **/
  -- /*****************************************/
  % INCLUDE common_declaration_area;
  /* variables common to PROGRAM1, PROGRAM2 and external procedure
DELETE_TS */
  DCL DELETE_TS EXTERNAL ENTRY;
  DCL DATETIME BUILTIN;
  DCL input_ts_written BIN FIXED(15) INIT(-1);
                                          /* Ø-successfully written */
  DCL convid_name CHAR(4); /* variables necessary to APPC commands  */
  DCL current_conv_state BIN FIXED(31);
  /********************** program ************************/
  EXEC CICS SYNCPOINT;
  CALL RECEIVE_INPUT_STRING;
  --  /* logical flow of program   */
  IF input_ts_written=Ø
  THEN CALL WRITE_OUTPUT_STRING_TS(check_point_number);
  /* check_point_number=1 */
  --
  IF input_ts_written=Ø
  THEN CALL WRITE_OUTPUT_STRING_TS(check_point_number);
  /* check-point number=2 */
  --
  IF input_ts_written=Ø
  THEN CALL WRITE_OUTPUT_STRING_TS(check_point_number);
  /* check-point number=n */
  ---
  CALL SEND_OUTPUT_STRING;
  /***************************** procedures ***************************/
  RECEIVE_INPUT_STRING: PROC;
    /*****************************************************/
    /* receives input string sent from client or PROGRAM2 */
    /*****************************************************/
    IF EIBCALEN>Ø THEN
    DO;
      input_string = commarea_input_string;
      started_from_tag='p';
      CALL WRITE_INPUT_STRING_TS;
    END;
    ELSE DO;
      started_from_tag='c';
      EXEC CICS ASSIGN FACILITY(convid_name) RESP(responce_value);
/*  returns a 4-byte identifier of the facility that initiated the
```

```
transaction */
      EXEC CICS RECEIVE CONVID(convid_name) INTO(input_string)
      STATE(current_conv_state)  LENGTH(input_string_length)
      MAXLENGTH(mis_length) RESP(responce_value);
      /*************************************************/
      /* Receives data on an APPC mapped conversation.    */
      /* Length of input string must not be greater than  */
      /* maximum_input_string_length, so MAXLENGTH option */
      /* is used.                                         */
      /*************************************************/
      IF responce_value=Ø THEN CALL WRITE_INPUT_STRING_TS;
    END;
  END RECEIVE_INPUT_STRING;

  WRITE_INPUT_STRING_TS: PROC;
    /*****************************************************************/
    /* If input_string_ts exists: write next item into it, check   */
    /* whether it exceeds max_allowed_number_of_item_in_ts and,    */
    /* if it does, call DELETE_TS procedure. If it does not exist, */
    /* write the first item in it.                                  */
    /*****************************************************************/
    EXEC CICS READQ TS QUEUE(input_string_ts_name) INTO(input_string_ts)
    NUMITEMS(current_number_of_item_in_ts) RESP(responce_value);
    IF responce_value¬=Ø
    THEN current_number_of_item_in_ts=1;
    ELSE DO;
      current_number_of_item_in_ts = current_number_of_item_in_ts+1;
      IF current_number_of_item_in_ts > max_allowed_number_of_item_in_ts
THEN
      DO;
        CALL DELETE_TS(current_number_of_item_in_ts);
        current_number_of_item_in_ts = current_number_of_item_in_ts+1;
      END;
    END;
    input_string_ts.current_datetime=DATETIME;
    EXEC CICS WRITEQ TS QUEUE(input_string_ts_name)
FROM(input_string_ts)
    ITEM(current_number_of_item_in_ts) MAIN RESP(input_ts_written);
  END WRITE_INPUT_STRING_TS;

  WRITE_OUTPUT_STRING_TS: PROC(check_point_number);
    /*****************************************************************/
    /* First call of this procedure writes item into ts, subsequent  */
    /* calls rewrite the same item in order to trace the point of a  */
    /* possible abend, final call from SEND_OUTPUT_STRING procedure  */
    /* has check_point_number=Ø and rewrites item with the answer    */
    /* output string successfully sent to the client.               */
    /*****************************************************************/
    output_string_ts.current_datetime=DATETIME;
    IF check_point_number =1
```

```
  THEN EXEC CICS WRITEQ TS QUEUE(output_string_ts_name)
       FROM(output_string_ts)
       ITEM(current_number_of_item_in_ts) MAIN RESP(responce_value);
  ELSE EXEC CICS WRITEQ TS QUEUE(output_string_ts_name)
       FROM(output_string_ts)
       ITEM(current_number_of_item_in_ts) REWRITE MAIN
       RESP(responce_value);
  IF responce_value¬=Ø THEN;
END WRITE_OUTPUT_STRING_TS;


SEND_OUTPUT_STRING: PROC;
   /*****************************************************************/
  /* Sends answer output string to the client (and rewrites it     */
  /* into output_string_ts) or returns control to PROGRAM2, and    */
  /* (optionally) restores all the changes depending on whether it */
  /* is allowed to end PROGRAM1 started from PROGRAM2 by saving    */
  /* all the changes.                                              */
   /*****************************************************************/
  IF EIBCALEN=Ø THEN
  DO;
    EXEC CICS SEND CONVID(convid_name) FROM(output_string) LAST
    LENGTH(output_string_length)  STATE(current_conv_state) CONFIRM
    RESP(responce_value);
    IF responce_value¬=Ø THEN
    DO;     /** output string could not be sent **/
      EXEC CICS SYNCPOINT ROLLBACK;
      EXEC CICS ISSUE ABEND CONVID(convid_name);
      EXEC CICS RETURN;
    END;
    IF input_ts_written=Ø
    THEN CALL WRITE_OUTPUT_STRING_TS(check_point_number);
    /* check-point number=Ø, last rewrite */
    EXEC CICS FREE CONVID(convid_name) STATE(current_conv_state);
    /* frees the session */
    EXEC CICS RETURN;
  END;
  ELSE DO;
    EXEC CICS SYNCPOINT ROLLBACK; /* optional */
    IF input_ts_written=Ø
    THEN CALL WRITE_OUTPUT_STRING_TS(check_point_number);
    /* check-point number=Ø */
    commarea_record='';
    commarea_flag=4;
    EXEC CICS XCTL PROGRAM('PROGRAM2')
    COMMAREA(commarea_record)  RESP(responce_value);
    IF responce_value¬=Ø THEN
    DO;  /** return to CICS with appropriate message **/
message=' Unable to return to PROGRAM2. Reason '||pic(responce_value);
      EXEC CICS SEND FROM(message);
```

```
        EXEC CICS RETURN;
      END;
    END;
  END SEND_OUTPUT_STRING;

END PROGRAM1;
```

## PROGRAM2 (connected to a transaction TRN2 with map MAP2) code:

```
PROGRAM2: PROC(PTR1) OPTIONS(MAIN);
  % INCLUDE common_declaration_area;
  % INCLUDE MAP2;      /* PL/1 declarations of MAP2 variables */
  % INCLUDE DFHAID;    /* function keys handle */
  % INCLUDE DFHBMSCA;  /* bms support */
  DCL DELETE_TS EXTERNAL ENTRY;

DCL input_string_cursor_in_positions(6) BIN FIXED(15)
INIT(259,499,739,979,1219,1459);
/* predestined possible cursor positions next to input string */
  DCL output_string_cursor_in_positions(6) BIN FIXED(15)
  INIT(339,579,819,1Ø59,1299,1539);
/* predestined possible cursor positions next to output string */

  DCL all_strings_ts CHAR(8);
  DCL screen CHAR(192Ø);
  /***************** program ******************/
  IF EIBCALEN=Ø THEN
  DO;  /* first time in program */
    EXEC CICS GETMAIN SET(PTR1) FLENGTH(STG(commarea_record));
    commarea_record=''; /* initializes variables in commarea_record */
  END;
  all_strings_ts=EIBTRMID||EIBTRNID; /* unique ts name */
  IF commarea_flag=Ø | commarea_flag=4 THEN
  DO;  /* first time in PROGRAM2 or return from PROGRAM1 */
    CALL  FILL_ALL_STRINGS_TS;
    commarea_flag=1;
    CALL SEND_MAP_AND_RETURN_TO_TRAN;
  END;
  IF commarea_flag>1 THEN
  DO;
     /*****************************************************************/
    /* PF6 pressed: if input_string has been displayed onto blank    */
    /* screen and ENTER is pressed  PROGRAM1 is started with possibly*/
    /* changed input string, else it displays current map.           */
     /*****************************************************************/
    IF EIBAID=DFHENTER & commarea_flag=2  THEN
    DO;
       /*****************************************************************/
      /* Rewrites commarea_input_string with current content of the  */
```

```
      /* screen. It allows alternation of the input string,          */
      /* restriction being that first 4 characters are 'TRN1', and   */
      /* length is limited to mis_length. If the lengerr condition   */
      /* occurs (response_value=22) it will be ignored.              */
       /*************************************************************/
EXEC CICS RECEIVE INTO(commarea_input_string)  LENGTH(mis_length)
RESP(response_value);
IF (response_value¬=Ø & response_value¬=22) | (response_value=Ø &
      SUBSTR(commarea_input_string,  1, 4)¬='TRN1') THEN
      DO;
        EXEC CICS READQ TS QUEUE(all_strings_ts) INTO(MAP2O)
        ITEM(commarea_current_item_in_all_strings_ts);
        commarea_flag=1;
        IF response_value¬=Ø & response_value¬=22
  THEN messageo='RECEIVE ERROR: '||pic(response_value);
  ELSE messageo='WRONG FIRST 4 CHARACTERS OF INPUT STRING '||
  SUBSTR(commarea_input_string,  1, 4);
        CALL SEND_MAP_AND_RETURN_TO_TRAN;
END;
      CALL XCTL_PROGRAM1;
    END;
    EXEC CICS READQ TS QUEUE(all_strings_ts) INTO(MAP2O)
    ITEM(commarea_current_item_in_all_strings_ts);
    commarea_flag=1;
    CALL SEND_MAP_AND_RETURN_TO_TRAN;
  END;

  EXEC CICS RECEIVE MAP('MAP2') MAPSET('MAP2') RESP(responce_value);
  IF responce_value¬=Ø THEN;

  SELECT (EIBAID);
    WHEN(DFHENTER) DO;
      /* puts 'choosen' string into commarea_input_string and pass it to
         PROGRAM1 */
      CALL  CHECK_CURSOR_POSITION_AND_READ_TS;
      CALL XCTL_PROGRAM1;
    END;

    WHEN(DFHPF6) DO;
      /*************************************************************/
     /* Puts 'choosen' string into commarea_input_string and sends  */
     /* it to blank screen. Presumptions is that both strings fit   */
     /* into screen size (less than 192Ø characters), otherwise     */
     /* another temporary storage must have been created with       */
     /* capability of scrolling it forward and backward. In order to */
     /* simplify the matter input and output strings were limited   */
     /* in this example.                                            */
     /* The user of transaction is responsible for altering the     */
     /* input string correctly.                                     */
      /*************************************************************/
```

```
      CALL  CHECK_CURSOR_POSITION_AND_READ_TS;
      screen=commarea_input_string;
      EXEC CICS SEND FROM(screen) ERASE;
      EXEC CICS RETURN COMMAREA(commarea_record) TRANSID('TRN2');
    END;

    WHEN(DFHPF8) DO;
                 /* shows next map(item) of input and output strings */
      EXEC CICS READQ TS QUEUE(all_strings_ts) INTO(MAP2O)
      NUMITEMS(number_of_items_in_ts);
      commarea_current_item_in_all_strings_ts=
      commarea_current_item_in_all_strings_ts+1;
      IF commarea_current_item_in_all_strings_ts> number_of_items_in_ts
    THEN commarea_current_item_in_all_strings_ts= number_of_items_in_ts;
      EXEC CICS READQ TS QUEUE(all_strings_ts) INTO(MAP2O)
      ITEM(commarea_current_item_in_all_strings_ts);
      IF commarea_current_item_in_all_strings_ts= number_of_items_in_ts
      THEN messageo='Last item displayed';
      ELSE messageo=' ';
      CALL SEND_MAP_AND_RETURN_TO_TRAN;
    END;

    WHEN(DFHPF7) DO;
                /* shows previous screen of input and output strings */
      commarea_current_item_in_all_strings_ts=
      commarea_current_item_in_all_strings_ts-1;
      IF commarea_current_item_in_all_strings_ts< 1
      THEN  commarea_current_item_in_all_strings_ts= 1;
      EXEC CICS READQ TS QUEUE(all_strings_ts) INTO(MAP2O)
      ITEM(commarea_current_item_in_all_strings_ts);
      IF commarea_current_item_in_all_strings_ts= 1
      THEN messageo='First item displayed';
      ELSE messageo=' ';
      CALL SEND_MAP_AND_RETURN_TO_TRAN;
    END;

    WHEN(DFHPF1Ø) DO;  /* calls DELETE_TS external procedure */
    EXEC CICS READQ TS QUEUE(input_string_ts_name) INTO(input_string_ts)
      NUMITEMS(current_number_of_item_in_ts) RESP(responce_value);
         IF responce_value=Ø THEN
      DO;
        CALL DELETE_TS(current_number_of_item_in_ts);
         CALL FILL_ALL_STRINGS_TS;
  messageo='Temporary storages deleted, left data displayed.';
      END;
CALL  SEND_MAP_AND_RETURN_TO_TRAN;
    END;

    WHEN(DFHPF12) DO;
                 /* unconditionally deletes all temporary storages **/
```

```
  EXEC CICS DELETEQ TS QUEUE(input_string_ts_name) RESP(responce_value);
      IF responce_value¬=Ø THEN;
 EXEC CICS DELETEQ TS QUEUE(output_string_ts_name) RESP(responce_value);
      IF responce_value¬=Ø THEN;
      EXEC CICS DELETEQ TS QUEUE(all_strings_ts) RESP(responce_value);
      IF responce_value¬=Ø THEN;
      messageo='Temporary storages deleted.'||
              ' There is no data to test and trace';
      CALL SEND_MAP_AND_RETURN_TO_TRAN;
    END;

    WHEN(PF3) DO; /* exit to CICS */
      DELETEQ TS QUEUE(all_strings_ts) RESP(responce_value);
      IF responce_value¬=Ø THEN;
      message='  You have finished work with transaction TRN2';
      EXEC CICS SEND FROM(message) ERASE;
      EXEC CICS RETURN;
    END;

    OTHER  CALL SEND_MAP_AND_RETURN_TO_TRAN;

  END; /* SELECT(EIBAID) */
  /*************** procedures ******************/
  XCTL_PROGRAM1: PROC;
    DELETEQ TS QUEUE(all_strings_ts) RESP(responce_value);
    IF responce_value¬=Ø THEN;
    /* All_strings_ts is deleted to free the memory */
    EXEC CICS XCTL PROGRAM('PROGRAM1') COMMAREA(commarea_record)
    RESP(responce_value);
    IF responce_value¬=Ø THEN
    DO;  /** exit to CICS with appropriate message **/
      message=' Unable to return to PROGRAM2. Reason '
      ||pic(responce_value);
      EXEC CICS SEND FROM(message);
      EXEC CICS RETURN;
    END;
  END XCTL_PROGRAM1;

  CHECK_CURSOR_POSITION_AND_READ_TS: PROC;
     /*****************************************************************/
    /* If cursor is on predefined "cursor in Pos" field next to      */
    /* input or output string, and item number is in field           */
    /* itemnumo(i) procedure reads the item and puts i/o-string      */
    /* into commarea_input_string                                    */
     /*****************************************************************/
    DCL item_found BIT(1) INIT('Ø'B);
    LOOP: DO i=1 TO 6;
IF (EIBCPOSN=input_string_cursor_positions(i) |
(EIBCPOSN=output_string_cursor_positions(i) & EIBAID=DFHPF6))
& itemnumo(i)>Ø THEN
```

```
    DO;
      item_found='1'B;
      current_number_of_item_in_ts=itemnumo(i);
      IF EIBCPOSN= input_string_cursor_positions(i) THEN
      DO;
        commarea_flag=2; /* input_string 'positioned' */
        EXEC CICS READQ TS QUEUE(input_string_ts_name)
        INTO(input_string_ts) ITEM(current_number_of_item_in_ts);
        commarea_input_string=input_string;
      END;
      ELSE DO;
        commarea_flag=3; /* output_string 'positioned' */
        EXEC CICS READQ TS QUEUE(output_string_ts_name)
        INTO(output_string_ts) ITEM(current_number_of_item_in_ts);
        commarea_input_string=output_string;
      END;
      LEAVE LOOP;
    END;
  END LOOP;
  IF item_found ='Ø'B
  THEN CALL SEND_MAP_AND_RETURN_TO_TRAN;
END CHECK_CURSOR_POSITION_AND_READ_TS: PROC;

FILL_ALL_STRINGS_TS: PROC;
   /*****************************************************************/
  /* Reads all input/output_string_ts' items and puts its contents */
  /* to a predetermined fields on map, and each filled map kept as */
  /* one item in all_strings_ts. Called on returning from PROGRAM1 */
  /* it displays the last map, otherwise displays the first one.   */
  /* Like input and output, all_strings_ts must not be defined as  */
  /* recoverable because WRITEQ and DELETEQ are within the same    */
  /* unit of work.                                                 */
   /*****************************************************************/
  DCL line_on_map BIN FIXED(15) INIT(Ø);
  EXEC CICS DELETEQ TS QUEUE(all_strings_ts) RESP(responce_value);
  IF responce_value¬=Ø THEN;
  commarea_current_item_in_all_strings_ts=Ø;
  i=1;
  EXEC CICS READQ TS QUEUE(input_string_ts_name) INTO(input_string_ts)
  NUMITEMS(number_of_item_in_ts) RESP(responce_value);
  IF responce_value¬=Ø THEN
  DO;
    messageo=' There is no data to test and trace';
    CALL SEND_MAP_AND_RETURN_TO_TRAN;
  END;
  DO i=1 to number_of_item_in_ts;
    line_on_map= line_on_map +1;
    IF line_on_map =7 THEN
    DO;  /** filled map **/
      commarea_current_item_in_all_strings_ts=
```

```
      commarea_current_item_in_all_strings_ts+1;
      EXEC CICS WRITEQ TS QUEUE(all_strings_ts) FROM(MAP2O)
ITEM(commarea_current_item_in_all_strings_ts);
      line_on_map =1;
    END;
  EXEC CICS READQ TS QUEUE(input_string_ts_name) INTO(input_string_ts)
    ITEM(i) RESP(responce_value);
    IF responce_value=Ø THEN
    DO;
      inputsto(line_on_map)=input_string;
      inputdto(line_on_map)= input_string_ts.current_datetime;
      itemnumo(line_on_map)=i;
      fromtago(line_on_map)=started_from_tag;
EXEC CICS READQ TS QUEUE(output_string_ts_name) INTO(output_string_ts)
      ITEM(i) RESP(responce_value);
      IF responce_value=Ø THEN
      DO;
        outputsto(line_on_map)= check_point_number||output_string;
        outputdto(line_on_map)= output_string_ts.current_datetime;
      END;
      ELSE DO;
  outputsto(line_on_map)= ' ';
        outputdto(line_on_map)= ' ';
      END;
    END;
  END;
  IF line_on_map <6 THEN
  DO;  /** last map **/
    commarea_current_item_in_all_strings_ts=
    commarea_current_item_in_all_strings_ts+1;
    DO i=line_on_map TO 6;  /* clears rest of lines in map */
      inputsto(line_on_map)=' ';
      inputdto(line_on_map)= ' ';
      itemnumo(line_on_map)=Ø;
      fromtago(line_on_map)=' ';
      outputsto(line_on_map)=' ';
      outputdto(line_on_map)= ' ';
    END;
    EXEC CICS WRITEQ TS QUEUE(all_strings_ts) FROM(MAP2O)
    ITEM(commarea_current_item_in_all_strings_ts);
  END;
  IF commarea_flag<=1 THEN commarea_current_item_in_all_strings_ts=1;
  EXEC CICS READQ TS QUEUE(all_strings_ts) INTO(MAP2O)
  ITEM(commarea_current_item_in_all_strings_ts);
END FILL_ALL_STRINGS_TS;

SEND_MAP_AND_RETURN_TO_TRAN: PROC;
IF commarea_flag=Ø | commarea_flag=4 | EIBAID=DFHPF1Ø | EIBAID=DFHPF12
  THEN EXEC CICS SEND MAP('MAP2') MAPSET('MAP2') CURSOR ERASE FREEKB;
ELSE EXEC CICS SEND MAP('MAP2') MAPSET('MAP2') CURSOR DATAONLY FREEKB;
```

```
    EXEC CICS RETURN COMMAREA(commarea_record) TRANSID('TRN2');
  END SEND_MAP_AND_RETURN_TO_TRAN;

END PROGRAM2;

/*********************************************************/
/* common_declaration_area, description of variables used */
/* by PROGRAM1, PROGRAM2 and DELETE_TS procedures         */
/*********************************************************/
DCL input_string_ts_name    CHAR(8) INIT('TRN1INPU');
DCL output_string_ts_name   CHAR(8) INIT('TRN1OUTP');
DCL maximum_input_string_length   BIN FIXED(15) INIT(mis_length);
/*********************************************************************/
/* limited to (32763(limited length of item in ts)-23)=32740 bytes */
/*********************************************************************/
DCL maximum_output_string_length  BIN FIXED(15) INIT(mos_length);
/*****************************************/
/* limited to (32763-22-cpn_length) bytes  */
/*****************************************/
DCL input_string_length  BIN FIXED(15);
DCL max_allowed_number_of_item_in_ts BIN FIXED(15) INIT(mano);
DCL current_number_of_item_in_ts BIN FIXED(15);
DCL 1 input_string_ts,   /* input temporary storage record */
      2 picture(current_number_of_item_in_ts) PIC '(5)9',
      2 current_datetime CHAR(17),
      2 input_string CHAR(mis_length)),
     /* first 4 characters of input_string are 'TRN1' others are data */
      2 started_from_tag CHAR(1); /* 'c' client, 'p' PROGRAM2 */
DCL 1 output_string_ts,  /* output temporary storage record */
      2 picture(current_number_of_item_in_ts) PIC '(5)9',
      2 current_datetime CHAR(17),
      2 check_point_number PIC(cpn_length),
         2 output_string CHAR(mos_length));
DCL responce_value BIN FIXED(31);
DCL message CHAR(message_length);
DCL number_of_items_in_ts BIN FIXED(15);
DCL i BIN FIXED(15);
DCL (SUBSTR, STG, CSTG, ADDR) BUILTIN;
/* commarea_description */
DCL PTR1 POINTER;
DCL 1 commarea_record,
2 commarea_flag DEC FIXED(1), /* Necessary to PROGRAM2. 0-first entry;
1-     other entries; 2,3 F6 pressed; 4-returning from PROGRAM1 */
      2 commarea_current_item_in_all_strings_ts BIN FIXED(15),
2 commarea_input_string  CHAR(mis_length);

/*******************************************************/
/* external procedure called from PROGRAM1 and PROGRAM2 */
/*******************************************************/
DELETE_TS:  PROC(current_number_of_item_in_ts);
```

```
 /*****************************************************************/
/* Main temporary storage is in fact VSAM file which CICS updates  */
/* with every new item, so it is recommendable (and more auditable */
/* to PROGRAM2) to limit the number of records stored in it, up to */
/* the value of max_allowed_number_of_item_in_ts variable (mano).  */
/* If number of items exceeds "mano" value, all items in           */
/* input_string_ts which 'pair' with  output_string_ts has not     */
/* reached last check_point_number (value  = Ø) will be kept along */
/* with its pair in output_string_ts. If there are no pairs at all,*/
/* both (input/output) temporary storages will be deleted.         */
/* Procedure returns number of kept items in                       */
/*current_number_of_item_in_ts variable.                           */
/* Restriction: queues must not be defined as recoverable because  */
/* EXEC CICS WRITEQ command follows DELETEQ within the same unit   */
/* of work, which is not allowed for such queues by definition.    */
 /*****************************************************************/
% INCLUDE common_declaration_area;
                                 /* excluding commarea_description */
DCL 1 temporary_input_string_ts (mano),
      2 picture(current_number_of_item_in_ts) PIC '(5)9',
      2 current_datetime CHAR(17),
      2 input_string CHAR(mis_length),
      2 started_from_tag CHAR(1);
DCL 1 temporary_output_string_ts (mano),
      2 picture(current_number_of_item_in_ts) PIC '(5)9',
      2 current_datetime CHAR(17),
      2 check_point_number PIC(cpn_length),
         2 output_string CHAR(mos_length);
DCL (j,number_of_kept) BIN FIXED(15) INIT(Ø);

temporary_output_string_ts(*)='';
current_number_of_item_in_ts=1;
EXEC CICS READQ TS QUEUE(input_string_ts_name) INTO(input_string_ts)
ITEM(current_number_of_item_in_ts) NUMITEMS(number_of_items_in_ts)
RESP(responce_value);
DO WHILE(current_number_of_item_in_ts <= number_of_items_in_ts);
  IF respoence_value=Ø THEN
  DO;
    EXEC CICS READQ TS QUEUE(output_string_ts_name)
    INTO(output_string_ts) ITEM(current_number_of_item_in_ts)
    RESP(responce_value);
    IF responce_value¬=Ø | output_string_ts.check_point_number¬=Ø THEN
    DO;  /** abend **/
       number_of_kept=number_of_kept+1;
  temporary_input_string_ts(number_of_kept)= input_string_ts, by name;
      IF responce_value=Ø
      THEN temporary_output_string_ts(number_of_kept)=
           output_string_ts, by name;
    END;
  END;
```

```
    current_number_of_item_in_ts= current_number_of_item_in_ts+1;
    EXEC CICS READQ TS QUEUE(input_string_ts_name)
    INTO(input_string_ts) ITEM(current_number_of_item_in_ts)
    RESP(responce_value);
  END /* DO WHILE */
  EXEC CICS DELETEQ TS QUEUE(input_string_ts_name) RESP(responce_value);
  EXEC CICS DELETEQ TS QUEUE(output_string_ts_name)
RESP(responce_value);
  IF number_of_kept>Ø THEN
  DO j=1 to number_of_kept;
    EXEC CICS WRITEQ TS QUEUE(input_string_ts_name)
    FROM(temporary_input_string_ts(j))
    ITEM(j) MAIN RESP(responce_value);
    IF temporary_output_string_ts(j) ¬=''
    THEN EXEC CICS WRITEQ TS QUEUE(output_string_ts_name)
         FROM(temporary_output_string_ts(j))
         ITEM(j) MAIN RESP(responce_value);
  END;
  current_number_of_item_in_ts= number_of_kept;
END DELETE_TS;
```

## MAP2 Assembler code:

```
         PRINT ON,NOGEN
MAP2     DFHMSD TYPE=MAP,LANG=PLI,MODE=INOUT,STORAGE=AUTO,SUFFIX=
MAP2     DFHMDI SIZE=(24,8Ø),COLUMN=1,LINE=1,DATA=FIELD,TIOAPFX=YES,   *
               OBFMT=NO
         DFHMDF POS=(1,19),LENGTH=23,INITIAL='Test and trace PROGRAM1',*
               ATTRB=(PROT,NORM)
         DFHMDF POS=(2,15),LENGTH=6,INITIAL='cursor',ATTRB=(PROT,NORM)
         DFHMDF POS=(2,69),LENGTH=7,INITIAL='started',ATTRB=(PROT,NORM)
         DFHMDF POS=(3,1),LENGTH=8,INITIAL='datetime',ATTRB=(PROT,NORM)
         DFHMDF POS=(3,15),LENGTH=42,                                  *
               INITIAL='on Pos strings (input above, output below)',   *
               ATTRB=(PROT,NORM)
         DFHMDF POS=(3,69),LENGTH=8,INITIAL='from tag',ATTRB=(PROT,NORM*
               )
* INPUTDT                          INPUTDT
INPUTDT  DFHMDF POS=(3,8Ø),LENGTH=17,ATTRB=(PROT,NUM,NORM,FSET)
         DFHMDF POS=(4,18),LENGTH=Ø,ATTRB=(PROT,NORM)
* INPUTDT                          INPUTDT
DFHØØØ1  DFHMDF POS=(6,8Ø),LENGTH=17,ATTRB=(PROT,NUM,NORM,FSET)
         DFHMDF POS=(7,18),LENGTH=Ø,ATTRB=(PROT,NORM)
* INPUTDT                          INPUTDT
DFHØØØ2  DFHMDF POS=(9,8Ø),LENGTH=17,ATTRB=(PROT,NUM,NORM,FSET)
         DFHMDF POS=(1Ø,18),LENGTH=Ø,ATTRB=(PROT,NORM)
* INPUTDT                          INPUTDT
DFHØØØ3  DFHMDF POS=(12,8Ø),LENGTH=17,ATTRB=(PROT,NUM,NORM,FSET)
         DFHMDF POS=(13,18),LENGTH=Ø,ATTRB=(PROT,NORM)
* INPUTDT                          INPUTDT
```

```
DFH0004  DFHMDF POS=(15,80),LENGTH=17,ATTRB=(PROT,NUM,NORM,FSET)
         DFHMDF POS=(16,18),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTDT                          INPUTDT
DFH0005  DFHMDF POS=(18,80),LENGTH=17,ATTRB=(PROT,NUM,NORM,FSET)
         DFHMDF POS=(19,18),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTPOS                         INPUTPOS
INPUTPO  DFHMDF POS=(4,19),LENGTH=1,ATTRB=(UNPROT,NORM,FSET)
         DFHMDF POS=(4,21),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTPOS                         INPUTPOS
DFH0006  DFHMDF POS=(7,19),LENGTH=1,ATTRB=(UNPROT,NORM,FSET)
         DFHMDF POS=(7,21),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTPOS                         INPUTPOS
DFH0007  DFHMDF POS=(10,19),LENGTH=1,ATTRB=(UNPROT,NORM,FSET)
         DFHMDF POS=(10,21),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTPOS                         INPUTPOS
DFH0008  DFHMDF POS=(13,19),LENGTH=1,ATTRB=(UNPROT,NORM,FSET)
         DFHMDF POS=(13,21),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTPOS                         INPUTPOS
DFH0009  DFHMDF POS=(16,19),LENGTH=1,ATTRB=(UNPROT,NORM,FSET)
         DFHMDF POS=(16,21),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTPOS                         INPUTPOS
DFH0010  DFHMDF POS=(19,19),LENGTH=1,ATTRB=(UNPROT,NORM,FSET)
         DFHMDF POS=(19,21),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTST                          INPUTST
INPUTST  DFHMDF POS=(4,22),LENGTH=45,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(4,68),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTST                          INPUTST
DFH0011  DFHMDF POS=(7,22),LENGTH=45,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(7,68),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTST                          INPUTST
DFH0012  DFHMDF POS=(10,22),LENGTH=45,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(10,68),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTST                          INPUTST
DFH0013  DFHMDF POS=(13,22),LENGTH=45,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(13,68),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTST                          INPUTST
DFH0014  DFHMDF POS=(16,22),LENGTH=45,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(16,68),LENGTH=0,ATTRB=(PROT,NORM)
* INPUTST                          INPUTST
DFH0015  DFHMDF POS=(19,22),LENGTH=45,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(19,68),LENGTH=0,ATTRB=(PROT,NORM)
* FROMTAG                          FROMTAG
FROMTAG  DFHMDF POS=(4,69),LENGTH=1,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(4,71),LENGTH=0,ATTRB=(PROT,NORM)
* FROMTAG                          FROMTAG
DFH0016  DFHMDF POS=(7,69),LENGTH=1,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(7,71),LENGTH=0,ATTRB=(PROT,NORM)
* FROMTAG                          FROMTAG
DFH0017  DFHMDF POS=(10,69),LENGTH=1,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(10,71),LENGTH=0,ATTRB=(PROT,NORM)
```

```
* FROMTAG                          FROMTAG
DFH0018  DFHMDF POS=(13,69),LENGTH=1,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(13,71),LENGTH=0,ATTRB=(PROT,NORM)
* FROMTAG                          FROMTAG
DFH0019  DFHMDF POS=(16,69),LENGTH=1,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(16,71),LENGTH=0,ATTRB=(PROT,NORM)
* FROMTAG                          FROMTAG
DFH0020  DFHMDF POS=(19,69),LENGTH=1,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(19,71),LENGTH=0,ATTRB=(PROT,NORM)
* ITEMNUM                          ITEMNUM
ITEMNUM  DFHMDF POS=(4,72),LENGTH=5,ATTRB=(PROT,NUM,DRK,FSET),          *
               PICIN='(5)9',PICOUT='(5)9'
         DFHMDF POS=(4,78),LENGTH=0,ATTRB=(PROT,NORM)
* ITEMNUM                          ITEMNUM
DFH0021  DFHMDF POS=(7,72),LENGTH=5,ATTRB=(PROT,NUM,DRK,FSET),          *
               PICIN='(5)9',PICOUT='(5)9'
         DFHMDF POS=(7,78),LENGTH=0,ATTRB=(PROT,NORM)
* ITEMNUM                          ITEMNUM
DFH0022  DFHMDF POS=(10,72),LENGTH=5,ATTRB=(PROT,NUM,DRK,FSET),         *
               PICIN='(5)9',PICOUT='(5)9'
         DFHMDF POS=(10,78),LENGTH=0,ATTRB=(PROT,NORM)
* ITEMNUM                          ITEMNUM
DFH0023  DFHMDF POS=(13,72),LENGTH=5,ATTRB=(PROT,NUM,DRK,FSET),         *
               PICIN='(5)9',PICOUT='(5)9'
         DFHMDF POS=(13,78),LENGTH=0,ATTRB=(PROT,NORM)
* ITEMNUM                          ITEMNUM
DFH0024  DFHMDF POS=(16,72),LENGTH=5,ATTRB=(PROT,NUM,DRK,FSET),         *
               PICIN='(5)9',PICOUT='(5)9'
         DFHMDF POS=(16,78),LENGTH=0,ATTRB=(PROT,NORM)
* ITEMNUM                          ITEMNUM
DFH0025  DFHMDF POS=(19,72),LENGTH=5,ATTRB=(PROT,NUM,DRK,FSET),         *
               PICIN='(5)9',PICOUT='(5)9'
         DFHMDF POS=(19,78),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUTDT                         OUTPUTDT
OUTPUTD  DFHMDF POS=(4,80),LENGTH=17,ATTRB=(PROT,NUM,NORM,FSET)
         DFHMDF POS=(5,18),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUTDT                         OUTPUTDT
DFH0026  DFHMDF POS=(7,80),LENGTH=17,ATTRB=(PROT,NUM,NORM,FSET)
         DFHMDF POS=(8,18),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUTDT                         OUTPUTDT
DFH0027  DFHMDF POS=(10,80),LENGTH=17,ATTRB=(PROT,NUM,NORM,FSET)
         DFHMDF POS=(11,18),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUTDT                         OUTPUTDT
DFH0028  DFHMDF POS=(13,80),LENGTH=17,ATTRB=(PROT,NUM,NORM,FSET)
         DFHMDF POS=(14,18),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUTDT                         OUTPUTDT
DFH0029  DFHMDF POS=(16,80),LENGTH=17,ATTRB=(PROT,NUM,NORM,FSET)
         DFHMDF POS=(17,18),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUTDT                         OUTPUTDT
DFH0030  DFHMDF POS=(19,80),LENGTH=17,ATTRB=(PROT,NUM,NORM,FSET)
```

```
         DFHMDF POS=(20,18),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUPOS                        OUTPUPOS
OUTPUPO  DFHMDF POS=(5,19),LENGTH=1,ATTRB=(UNPROT,NORM,FSET)
         DFHMDF POS=(5,21),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUPOS                        OUTPUPOS
DFH0031  DFHMDF POS=(8,19),LENGTH=1,ATTRB=(UNPROT,NORM,FSET)
         DFHMDF POS=(8,21),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUPOS                        OUTPUPOS
DFH0032  DFHMDF POS=(11,19),LENGTH=1,ATTRB=(UNPROT,NORM,FSET)
         DFHMDF POS=(11,21),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUPOS                        OUTPUPOS
DFH0033  DFHMDF POS=(14,19),LENGTH=1,ATTRB=(UNPROT,NORM,FSET)
         DFHMDF POS=(14,21),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUPOS                        OUTPUPOS
DFH0034  DFHMDF POS=(17,19),LENGTH=1,ATTRB=(UNPROT,NORM,FSET)
         DFHMDF POS=(17,21),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUPOS                        OUTPUPOS
DFH0035  DFHMDF POS=(20,19),LENGTH=1,ATTRB=(UNPROT,NORM,FSET)
         DFHMDF POS=(20,21),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUTST                        OUTPUTST
OUTPUTS  DFHMDF POS=(5,22),LENGTH=45,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(5,68),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUTST                        OUTPUTST
DFH0036  DFHMDF POS=(8,22),LENGTH=45,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(8,68),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUTST                        OUTPUTST
DFH0037  DFHMDF POS=(11,22),LENGTH=45,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(11,68),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUTST                        OUTPUTST
DFH0038  DFHMDF POS=(14,22),LENGTH=45,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(14,68),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUTST                        OUTPUTST
DFH0039  DFHMDF POS=(17,22),LENGTH=45,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(17,68),LENGTH=0,ATTRB=(PROT,NORM)
* OUTPUTST                        OUTPUTST
DFH0040  DFHMDF POS=(20,22),LENGTH=45,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(20,68),LENGTH=0,ATTRB=(PROT,NORM)
* MESSAGE                         MESSAGE
MESSAGE  DFHMDF POS=(21,80),LENGTH=77,ATTRB=(PROT,NORM,FSET)
         DFHMDF POS=(22,78),LENGTH=0,ATTRB=(PROT,NORM)
         DFHMDF POS=(23,1),LENGTH=78,                                  *
               INITIAL='ENTER+coP-start PROGRAM1, F7-previous, F8-next,*
                F3-exit, F12-delete ts (clear)',ATTRB=(PROT,NORM)
         DFHMDF POS=(24,1),LENGTH=69,                                  *
               INITIAL='F10-delete ts, F6+coP-full string show (+ alter*
                input string + ENTER)',ATTRB=(PROT,NORM)
         DFHMSD TYPE=FINAL
         END
```

The example below shows MAP2 with an example of a simple financial transaction called TRN1 that debits <account num>

with <amount> and checks <account balance>.

In the first pair of lines, PROGRAM1 ends regularly, but not expectedly, with <reason code>, which helps a CICS programmer to handle this situation and afterwards to start PROGRAM2 (second pair) where <account balance1> appears to be wrong. When <account balance1> is changed to the correct value <account balance2>, PROGRAM1 ends and returns <new account balance> (third pair) – all the changes were annulled with a rollback, since PROGRAM1 (TRN1) is started from PROGRAM2. Now the client starts PROGRAM1 with the same data and gets the same answer (fourth pair). In the fifth pair there is an example of an abend that a CICS programmer can analyse thanks to check_point_number=2 and <current content of output string>. Using these, the problem can be fixed, so that in the next try, with the same input string, it gets two steps further (sixth pair) etc.

```
 Test and trace PROGRAM1
               cursor                                                started
 datetime      on Pos strings (input above, output below)       from tag
20030527121276843     TRN1<account num1><account balance1><amount1>  c
20030527121291765     0database closed<reason code>

20030527122554435     TRN1<account num1><account balance1><amount1>  p
20030527122567536     0wrong account balance<account balance1>

20030527123265435     TRN1<account num1><account balance2><amount1>  p
20030527123266089     0<new account balance>

20030527123380163     TRN1<account num1><account balance2><amount1>  c
20030527123389361     0<new account balance>

20030527123498718     TRN1<account num2><account balance3><amount2>  c
20030527123499065     2<current content of output string>

20030527123506754     TRN1<account num3><account balance4><amount3>  p
20030527123507651     4<current content of output string>


 ENTER+coP-start PROGRAM1, F7-previous, F8-next, F3-exit, F12-delete ts
(clear)
 F10-delete ts, F6+coP-full string show (+ alter input string + ENTER)
```

*Vladimir Antonijevic*
*IT Analyst*
*Postal Savings Bank (Yugoslavia)*

# CICSPlex SM API – Assembler programs (command-level interface)

## 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.

The CICSPlex SM API has two interfaces: a command-level interface for programs written in Assembler, PL/I, COBOL, or C, and also a run-time interface, which supports programs written as REXX EXECs. My article in *CICS Update* 207 and 208 (February and March 2003) entitled *CICSPlex SM API – REXX EXECs (run-time interface)* concentrated on the run-time interface, and in that article I mentioned that a couple of the REXX EXECs were later re-written as Assembler programs. This article concentrates on those Assembler (command-level interface) programs. This was covered in the run-time interface article, but I would like to re-emphasize that the ability to specify a context, a scope within the context, and criteria within the scope are what makes the CICSPlex SM API so versatile and flexible.

The context is the name of a CICSPlex SM address space (CMAS) or CICSplex. If the context is a CICSplex, the scope further qualifies the context by specifying that the scope is the CICSplex 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 the PROGRAM resource PROGRAM=PROG1, and complex compound logical expressions, eg for the 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 COMMAND-LEVEL INTERFACE

The CICSPlex SM API command-level interface can be used in the following environments:

- CICS
- MVS batch
- MVS NetView
- MVS TSO

in programs written in the following languages:

- Assembler H Version 2 and later.
- OS PL/I Optimizing Compiler Version 2.3 and later.
- VS COBOL II Compiler Version 1.3.2 and later (excluding MVS NetView environment).
- C/370 Version 2.1 and later.

The command-level interface uses EXEC CPSM commands, which must be translated by the CICS translator in much the same way as EXEC CICS commands. The CICS translator

CPSM option must be specified. The language-specific macro library must be added to the SYSLIB concatenation for assembly/ compilation:

- SEYUMAC – Assembler
- SEYUPL1 – PL/I
- SEYUCOB – COBOL
- SEYUC370 – C/370.

Also the CPSM load library, SEYULOAD, must be added to the SYSLIB concatenation in the link-edit step to include the CICSPlex SM API stub:

```
INCLUDE SYSLIB(EYU9AMSI)
```

The command-level interface has two levels of command response codes, RESPONSE and REASON, similar to CICS RESP and RESP2. The CPSM API response codes can be tested in a similar way to CICS response codes, eg EYUVALUE(OK) is functionally equivalent to DFHVALUE(NORMAL) and indicates a normal or successful completion.

Please refer to the *CICSPlex SM Application Programming Guide* and *CICSPlex SM Application Programming Reference* for more detailed information regarding the CICSPlex SM API.

## MAINTENANCE

You should generally ensure that you are relatively up-to-date with maintenance. The programs in this article were written for CICS Transaction Server for OS/390 Version 1 Release 3 and therefore CICSPlex SM Version 1 Release 4. I opened two PMRs while testing the programs, which were more or less immediately accepted as APARs, and a third problem was discovered much later, but an APAR was accepted and finally resolved. If you are going to use these programs (or the REXX EXECs in the previous article) you will definitely need the following PTFs:

- APAR PQ54395, PTF UQ65622
- APAR PQ55708, PTF UQ61178
- APAR PQ77644, PTF UQ80451.

Equivalent APARs and PTFs for CICS Transaction Server for z/OS Version 2 Release 2 are available.

## BACKGROUND INFORMATION

Prior to CICSplex and dynamic transaction routing (using CPSM Workload Management) there was an application program that used the CICS SPI to inquire about tasks and transaction classes and based the decision on whether to start another transaction on that information. When the application was migrated to a CICSplex with multiple routing and application (or target) regions on multiple LPARs, there was a requirement to obtain the same information for all regions involved in the processing of that application from within a CICS transaction. The CICSPlex SM API was the most obvious and appropriate solution. Another alternative would have been the complete redesign of the original application.

The requirement was basically to make available to a CICS transaction information regarding tasks and transaction classes within a CICSplex, ie ideally a LINKable program. The application was interested not particularly in processing the data itself, but, rather, looking for true/false answers to requests, eg is transaction ABCD currently running anywhere or are transactions currently queueing in a transaction class?

## THE PROGRAMS

The Assembler programs CM431 and CM433 actually do the CPSM API processing. The COBOL programs CM430 and CM432 were written to test the respective Assembler programs and also to demonstrate how to use the CPSM API programs in COBOL programs.

- CM430 – COBOL/CICS main (test) program – transaction class resource.
- CM431 – Assembler/CICS subprogram – CPSM API TRANCLAS resource.
- CM431A01 – Assembler copybook COMMAREA.
- CM431A02 – Assembler copybook Temporary Storage Queue.
- CM431C01 – COBOL copybook COMMAREA.
- CM431C02 – COBOL copybook Temporary Storage Queue.
- CM432 – COBOL/CICS main (test) program – task resource.
- CM433 – Assembler/CICS subprogram – CPSM API TASK resource.
- CM433A01 – Assembler copybook COMMAREA.
- CM433A02 – Assembler copybook Temporary Storage Queue.
- CM433C01 – COBOL copybook COMMAREA.
- CM433C02 – COBOL copybook Temporary Storage Queue.

The following subprograms are used by CM431 and CM433. I did not want to rewrite the programs for this article so I have included the original subprograms, which I use in more or less all of my CICS programs:

- CM401 – write message to CICS Transient Data Queue (TDQ).
- CM401A01 – Assembler COMMAREA
- CM401C01 – COBOL COMMAREA
- CM402 – EXEC CICS diagnostics
- CM402A01 – Assembler COMMAREA
- CM402C01 – COBOL COMMAREA

- CM403 – write message to operator (SYSLOG)
- CM403A01 – Assembler COMMAREA
- CM403C01 – COBOL COMMAREA.

## CM430

The COBOL/CICS main program CM430 is essentially used to test the CM431 program. I have defined a transaction M430 for this program using the transaction profile DFHCICSP (UCTRAN=YES). The intention was to keep the program as open as possible, therefore the context, scope, and criteria can be specified by the user. Any criteria appropriate to the CPSM TRANCLAS resource table can be specified. If TRANCLAS resource table records matching the context, scope, and criteria specified are found they are written to a temporary storage queue, which will immediately be displayed using the CICS supplied CEBR transaction.

The following example requests information about any transaction classes named CARL in the CICSplex TPLEX:

```
m430 tplex tplex name=carl.

CONTEXT  : TPLEX
SCOPE    : TPLEX
CRITERIA : NAME=CARL.
NO. OF TRANCLAS RECORDS : 0000
PROCESSING COMPLETE. RC=0004
```

The following example requests information about transaction classes beginning with DFH and where no transactions are queued in region CICSTEST within CICSplex TPLEX:

```
m430 tplex cicstest "name=dfh* and queued=0."

 CEBR  TSQ CM430T282RESULTS SYSID TEST REC     1 OF  12    COL 1 OF   80
 ENTER COMMAND ===>
************************  TOP OF QUEUE  ******************************
0001 CICSTEST TRANCLAS(DFHEDFTC) MAX(0010) ACT(0000) QUE(0000) PUR(0000)
0002 CICSTEST TRANCLAS(DFHTCIND) MAX(0010) ACT(0000) QUE(0000) PUR(0000)
0003 CICSTEST TRANCLAS(DFHTCL01) MAX(0001) ACT(0000) QUE(0000) PUR(0000)
0004 CICSTEST TRANCLAS(DFHTCL02) MAX(0001) ACT(0000) QUE(0000) PUR(0000)
0005 CICSTEST TRANCLAS(DFHTCL03) MAX(0001) ACT(0000) QUE(0000) PUR(0000)
```

```
0006 CICSTEST TRANCLAS(DFHTCL04) MAX(0001) ACT(0000) QUE(0000) PUR(0000)
0007 CICSTEST TRANCLAS(DFHTCL05) MAX(0001) ACT(0000) QUE(0000) PUR(0000)
0008 CICSTEST TRANCLAS(DFHTCL06) MAX(0001) ACT(0000) QUE(0000) PUR(0000)
0009 CICSTEST TRANCLAS(DFHTCL07) MAX(0001) ACT(0000) QUE(0000) PUR(0000)
0010 CICSTEST TRANCLAS(DFHTCL08) MAX(0001) ACT(0000) QUE(0000) PUR(0000)
0011 CICSTEST TRANCLAS(DFHTCL09) MAX(0001) ACT(0000) QUE(0000) PUR(0000)
0012 CICSTEST TRANCLAS(DFHTCL10) MAX(0001) ACT(0000) QUE(0000) PUR(0000)
************************  BOTTOM OF QUEUE  ****************************



PF1 : HELP              PF2 : SWITCH HEX/CHAR     PF3 : TERMINATE BROWSE
PF4 : VIEW TOP          PF5 : VIEW BOTTOM         PF6 : REPEAT LAST FIND
PF7 : SCROLL BACK HALF    PF8 : SCROLL FORWARD HALF PF9 : VIEW RIGHT
PF10: SCROLL BACK FULL    PF11: SCROLL FORWARD FULL PF12: UNDEFINED
```

You should purge the TSQ before terminating the CEBR transaction.

```
       ****************************************************************
      *              C A R L   W A D E   M C B U R N I E               *
      *                - I T   C O N S U L T A N T -                   *
      *                        www.cwmit.com                           *
      * MODULE NAME = CM430                                            *
      * MODULE TYPE = COBOL/CICS Main Program                          *
      * DESCRIPTION = CICS CPSM TRANCLAS Test Program                  *
      *               This program produces TRANCLAS information       *
      *               based on the parameters entered by the user :-   *
      *               CONTEXT   - CPSM context for the request or      *
      *                           blank for default.                   *
      *               SCOPE     - CPSM scope for the request or        *
      *                           blank for default.                   *
      *               CRITERIA  - CPSM criteria for the request or     *
      *                           blank for default.                   *
      *                           If CRITERIA includes more then       *
      *                           one parameter, the parameters must   *
      *                           be enclosed by parentheses.          *
      *               This program is intended for use by CICS         *
      *               systems programmers, when debugging or testing   *
      *               the CICS CPSM API TRANCLAS Interface program     *
      *               (CM431).                                         *
           EJECT
      * CHANGE HISTORY:                                                *
           EJECT
      ***      I D E N T I F I C A T I O N    D I V I S I O N       ***
       IDENTIFICATION DIVISION.
       PROGRAM-ID. CM430.
           EJECT
      ***         E N V I R O N M E N T    D I V I S I O N          ***
```

```
 ENVIRONMENT DIVISION.
     EJECT
***                D A T A    D I V I S I O N                ***
 DATA DIVISION.
     EJECT
***     W O R K I N G - S T O R A G E    S E C T I O N       ***
 WORKING-STORAGE SECTION.
 77  W-RESP                              PIC S9(8) COMP.
 77  W-RESP2                             PIC S9(8) COMP.
 Ø1  WORKSTOR.
     Ø2  W-INPUT                          PIC X(1ØØ) VALUE SPACES.
     Ø2  W-OUTPUT.
     Ø5  W-OUTPUT-TXT                     PIC X(8Ø) VALUE SPACES.
     Ø5  W-OUTPUT-CTL                     PIC  X(1) VALUE X'13'.
     Ø2  W-TSQ.
     Ø5  W-PROGRAM                        PIC X(Ø5) VALUE SPACES.
     Ø5  W-TRMID                          PIC X(Ø4) VALUE SPACES.
     Ø5  W-REST                           PIC X(Ø7) VALUE SPACES.
     Ø2  W-NEXT-TRAN.
     Ø5  W-TRANSID                        PIC X(Ø4) VALUE 'CEBR'.
     Ø5  FILLER                           PIC X(Ø1) VALUE SPACES.
     Ø5  W-TSQN                           PIC X(16) VALUE SPACES.
     Ø2  W-UNSTRING.
     Ø5  W-TRAN                           PIC  X(4) VALUE SPACES.
     Ø5  W-CONTEXT                        PIC  X(8) VALUE SPACES.
     Ø5  W-SCOPE                          PIC  X(8) VALUE SPACES.
     Ø5  W-BEFORE-CRITERIA                PIC X(7Ø) VALUE SPACES.
     Ø5  W-CRITERIA                       PIC X(7Ø) VALUE SPACES.
     Ø5  W-CRITERIA-C                     PIC S9(3) VALUE Ø.
     Ø2  W-REGION                         PIC  X(8) VALUE SPACES.
     Ø2  W-TRANCLAS                       PIC  X(8) VALUE SPACES.
     Ø2  W-MAXACTIVE                      PIC  9999 VALUE Ø.
     Ø2  W-ACTIVE                         PIC  9999 VALUE Ø.
     Ø2  W-QUEUED                         PIC  9999 VALUE Ø.
     Ø2  W-PURGETHRESH                    PIC  9999 VALUE Ø.
     Ø2  W-RC                             PIC  9999 VALUE Ø.

     Ø2  W-RECORDS                        PIC  9999 VALUE Ø.
     Ø2  W-BLANK                          PIC  X(1) VALUE SPACES.
*-  Copybook - CM431 Communications Area                      -*
     COPY  CM431CØ1.
*-  Copybook - CM431 Temporary Storage Queue                  -*
     COPY  CM431CØ2.
     EJECT
***              L I N K A G E   S E C T I O N              ***
 LINKAGE SECTION.
     EJECT
***          P R O C E D U R E   D I V I S I O N            ***
 PROCEDURE DIVISION.
*
```

```
 MAINLINE.
*-  RECEIVE INPUT                                                -*
     EXEC CICS RECEIVE INTO (W-INPUT)
                       MAXLENGTH (LENGTH OF W-INPUT)
     END-EXEC.
*-  PARSE INPUT                                                  -*
     UNSTRING W-INPUT
         DELIMITED BY X'7D' OR X'7F'
         INTO W-BEFORE-CRITERIA
              W-CRITERIA COUNT IN W-CRITERIA-C
     END-UNSTRING.
     IF W-CRITERIA-C = Ø
        UNSTRING W-INPUT
            DELIMITED BY ',' OR ' '
            INTO W-TRAN
                 W-CONTEXT
                 W-SCOPE
                 W-CRITERIA COUNT IN W-CRITERIA-C
        END-UNSTRING
     ELSE
        UNSTRING W-BEFORE-CRITERIA
            DELIMITED BY ',' OR ' '
            INTO W-TRAN
                 W-CONTEXT
                 W-SCOPE
        END-UNSTRING
     END-IF.
*-  FORMAT MESSAGE - CONTEXT                                     -*
     STRING 'CONTEXT  : ' DELIMITED BY SIZE
            W-CONTEXT DELIMITED BY SIZE
       INTO W-OUTPUT-TXT.
*-  SEND MESSAGE - CONTEXT                                       -*
     EXEC CICS SEND
               FROM     (W-OUTPUT)
               LENGTH   (LENGTH OF W-OUTPUT)
               ERASE
     END-EXEC.
     MOVE SPACES              TO  W-OUTPUT-TXT.
*-  FORMAT MESSAGE - SCOPE                                       -*
     STRING 'SCOPE    : ' DELIMITED BY SIZE
            W-SCOPE DELIMITED BY SIZE
       INTO W-OUTPUT-TXT.
*-  SEND MESSAGE - SCOPE                                         -*
     EXEC CICS SEND
               FROM     (W-OUTPUT)
               LENGTH   (LENGTH OF W-OUTPUT)
               WAIT
     END-EXEC.
     MOVE SPACES              TO  W-OUTPUT-TXT.
*-  FORMAT MESSAGE - CRITERIA                                    -*
```

```
     STRING 'CRITERIA : ' DELIMITED BY SIZE
            W-CRITERIA DELIMITED BY SIZE
       INTO W-OUTPUT-TXT.
*-  SEND MESSAGE - CRITERIA                                    -*
     EXEC CICS SEND
               FROM     (W-OUTPUT)
               LENGTH   (LENGTH OF W-OUTPUT)
               WAIT
     END-EXEC.
     MOVE SPACES              TO  W-OUTPUT-TXT.
*-  SET UP OTHER REQUIRED PARAMETERS                           -*
     MOVE 'CM43Ø'             TO  W-PROGRAM.
     MOVE EIBTRMID            TO  W-TRMID.
     MOVE W-TSQ               TO  CM431COM-TSQ.
     MOVE W-CONTEXT           TO  CM431COM-CONTEXT.
     MOVE W-SCOPE             TO  CM431COM-SCOPE.
     MOVE W-CRITERIA          TO  CM431COM-CRITERIA.
     MOVE W-CRITERIA-C        TO  CM431COM-CRITERIA-L.
*-  LINK TO PROGRAM CM431 WITH COMMAREA.                       -*
     EXEC CICS LINK PROGRAM     ('CM431')
                    COMMAREA    (CM431COM)
                    LENGTH      (LENGTH OF CM431COM)
     END-EXEC.
*-  FORMAT MESSAGE - NO. OF RECORDS                            -*
     MOVE CM431COM-RECORDS    TO  W-RECORDS.
     STRING 'NO. OF TRANCLAS RECORDS : ' DELIMITED BY SIZE
            W-RECORDS DELIMITED BY SIZE
       INTO W-OUTPUT-TXT.
*-  SEND MESSAGE - NO. OF RECORDS                              -*
     EXEC CICS SEND
               FROM     (W-OUTPUT)
               LENGTH   (LENGTH OF W-OUTPUT)
               WAIT
     END-EXEC.
     MOVE SPACES              TO  W-OUTPUT-TXT.
*-  IF WE HAVE A TSQ TO PROCESS, PROCESS IT                    -*
     IF CM431COM-RECORDS NOT = Ø
        MOVE 'RESULTS'       TO  W-REST
        MOVE DFHRESP(NORMAL) TO  W-RESP
        PERFORM WITH TEST BEFORE UNTIL
                W-RESP NOT = DFHRESP(NORMAL)
          EXEC CICS READQ TS
                          QNAME        (CM431COM-TSQ)
                          INTO         (CM431TSQ)
                          LENGTH       (LENGTH OF CM431TSQ)
                          NEXT
                          RESP         (W-RESP)
                          RESP2        (W-RESP2)
          END-EXEC
*-  MOVE TRANCLAS INFORMATION TO DISPLAYABLE FIELDS            -*
```

```
          IF W-RESP = DFHRESP(NORMAL)
             MOVE CM431TSQ-REGION      TO  W-REGION
             MOVE CM431TSQ-TRANCLAS    TO  W-TRANCLAS
             MOVE CM431TSQ-MAXACTIVE   TO  W-MAXACTIVE
             MOVE CM431TSQ-ACTIVE      TO  W-ACTIVE
             MOVE CM431TSQ-QUEUED      TO  W-QUEUED
             MOVE CM431TSQ-PURGETHRESH TO  W-PURGETHRESH
*-  BUILD STRING FOR TSQ                                          -*
             STRING W-REGION DELIMITED BY SIZE
                    '  TRANCLAS(' DELIMITED BY SIZE
                    W-TRANCLAS DELIMITED BY SIZE
                    ')  MAX(' DELIMITED BY SIZE
                    W-MAXACTIVE DELIMITED BY SIZE
                    ')  ACT(' DELIMITED BY SIZE
                    W-ACTIVE DELIMITED BY SIZE
                    ')  QUE(' DELIMITED BY SIZE
                    W-QUEUED DELIMITED BY SIZE
                    ')  PUR(' DELIMITED BY SIZE
                    W-PURGETHRESH DELIMITED BY SIZE
                    ')' DELIMITED BY SIZE
               INTO W-OUTPUT-TXT
*-  WRITE STRING TO TSQ                                           -*
             EXEC CICS WRITEQ TS
                       QNAME        (W-TSQ)
                       FROM         (W-OUTPUT-TXT)
                       LENGTH       (LENGTH OF W-OUTPUT-TXT)
                       RESP         (W-RESP)
                       RESP2        (W-RESP2)
             END-EXEC
             MOVE SPACES  TO  W-OUTPUT-TXT
          END-IF
        END-PERFORM
*-  DELETE TSQ CREATED BY CM431                                   -*
        EXEC CICS DELETEQ TS
                          QNAME        (CM431COM-TSQ)
        END-EXEC
     END-IF.
     MOVE CM431COM-RC         TO  W-RC.
*-  FORMAT MESSAGE - PROCESSING COMPLETE                          -*
      STRING 'PROCESSING COMPLETE. RC=' DELIMITED BY SIZE
             W-RC DELIMITED BY SIZE
        INTO W-OUTPUT-TXT.
*-  SEND MESSAGE - PROCESSING COMPLETE                            -*
     EXEC CICS SEND
               FROM     (W-OUTPUT)
               LENGTH   (LENGTH OF W-OUTPUT)
               WAIT
     END-EXEC.
     MOVE SPACES              TO  W-OUTPUT-TXT.
     MOVE W-TSQ               TO  W-TSQN.
```

```
*-  IF WE HAVE CREATED A TSQ BROWSE IT WITH CEBR                -*
     IF CM431COM-RECORDS = Ø
        EXEC CICS RETURN
        END-EXEC
     ELSE
        EXEC CICS RETURN
                     TRANSID     (W-TRANSID)
                     INPUTMSG    (W-NEXT-TRAN)
                     INPUTMSGLEN (LENGTH OF W-NEXT-TRAN)
                     IMMEDIATE
        END-EXEC
     END-IF.
* C M 4 3 Ø    -  END                                              *
     EXIT PROGRAM.
```

## CM431

The CM431 program provides the CPSM API processing for the TRANCLAS (transaction class) resource table. The program contains defaults for context, scope, and criteria. The defaults for context and scope are based on a specific naming convention, whereby the first character of the CICS SYSID indicates the CICSplex to which the region belongs, eg a CICS region with a SYSID of TEST belongs to the CICSplex TPLEX, and a SYSID of PA01 is a region belonging to the CICSplex PPLEX. The advantage here is that the programmer wishing to call CM431 does not need to specify a context, ie he does not need to know in which CICSplex the program will run, which makes migration from one CICSplex to another far easier. The default criteria will obtain all transaction classes within the context and scope. You can, of course, change the defaults to meet your own requirements and naming conventions.

The CM431 program returns only specific fields from the TRANCLAS resource table and not all the fields available. Also, the table entries are provided only if the name of a temporary storage queue is specified in the COMMAREA field CM431COM_TSQ. This provides better performance for true/false type requests, eg if you wanted only to check that no transactions were being queued in transaction classes you could specify the criteria as QUEUED>0, and test for return code (CM431COM_RC) of 4 and 0 records

(CM431COM_RECORDS). The return code of 4 differentiates between zero records because of an error and zero records matching the context, scope, and criteria specified.

You could, of course, make a small amendment to the program to write the entire TRANCLAS record to the TSQ if required.

```
*ASM XOPTS(CICS SP CPSM)
CM431    TITLE 'CM431 : CPSM TRANCLAS RESOURCE'
*                 C A R L   W A D E   M C B U R N I E                 *
*                   -  I T   C O N S U L T A N T  -                   *
*                          www.cwmit.com                              *
* MODULE NAME = CM431                                                 *
* MODULE TYPE = CSECT  (Sub Program)                                  *
* DESCRIPTION = CICS CPSM API TRANCLAS Resource                       *
*               The program receives a communications area            *
*               CM431COM and uses the context, scope and criteria     *
*               specified in the communications area to obtain        *
*               information regarding defined CICS transaction        *
*               classes from CICSPlex SM.                             *
*               This information can additionally be stored in        *
*               a temporary storage queue mapped by CM431TSQ.         *
         EJECT
* CHANGE HISTORY:                                                     *
         EJECT
* REGISTER  EQUATES                      USAGE                        *
* REG  Ø    RØ                                                        *
* REG  1    R1                                                        *
* REG  2    R2        DSECT - CM431COM                                *
* REG  3    B1        Base Register for CSECT CM431                   *
* REG  4    R4        Work Register                                   *
* REG  5    R5        Work Register                                   *
* REG  6    R6        Work Register                                   *
* REG  7    R7        Work Register                                   *
* REG  8    R8        DSECT - CM4Ø1COM                                *
* REG  9    R9        DSECT - CM4Ø2COM                                *
* REG 1Ø    R1Ø       DSECT - CM4Ø3COM                                *
* REG 11    EIBREG    DSECT - DFHEIBLK                                *
* REG 12    B2        2nd Base Register for CSECT CM431               *
* REG 13    DYNREG    DSECT - DFHEISTG                                *
* REG 14    R14       Linkage                                         *
* REG 15    R15       Linkage                                         *
         EJECT
*- Copybooks                                                         -*
         COPY  CM4Ø1AØ1               DSECT - CM4Ø1COM  (TDQ MSG)
         EJECT
         COPY  CM4Ø2AØ1               DSECT - CM4Ø2COM  (EXEC DIAGS)
         EJECT
         COPY  CM4Ø3AØ1               DSECT - CM4Ø3COM  (WTO MSG)
```

```
         EJECT
         COPY  CM431A01                DSECT - CM431COM  (TRANCLAS)
         EJECT
         COPY  CM431A02                DSECT - CM431TSQ  (TRANCLAS)
         EJECT
         COPY  TRANCLAS                DSECT - TRANCLAS  (CPSM TABLE)
         EJECT
*- Addressability to DFHEISTG will be established by CICS.          -*
         DFHEISTG                      CICS DYNAMIC STORAGE
         EJECT
*- CM431 Dynamic Storage - Start                                    -*
DYNSTOR  DS    0H                      CM431 DYNAMIC STORAGE
SYSID       DS 0CL4                    CICS SYSID
CICSPLEX    DS CL1                     CICSPLEX PREFIX
REGION      DS CL1                     REGION TYPE
APPLICATION DS CL1                     APPLICATION
UNIQUE      DS CL1                     UNIQUE
DUMPCODE    DS  CL4                    DUMP CODE
RETC        DS  F                      RETURN CODE
Z401SR14    DS  F                      SAVE REGISTER 14
Z402_RESP1  DS  F                      RESP1 CODE
Z402_RESP2  DS  F                      RESP2 CODE
Z402SR14    DS  F                      SAVE REGISTER 14
Z403SR14    DS  F                      SAVE REGISTER 14
Z901SR14       DS  F                   SAVE REGISTER 14
Z901_D1        DS  D                   WORK DW
Z901_D2        DS  D                   WORK DW
*  Work areas for EXEC CPSM calls
Z901_RESPONSE  DS  F                   O - RECEIVES RESPONSE
Z901_REASON    DS  F                   O - RECEIVES REASON
Z901_THREAD    DS  F                   O - RECEIVES THREAD (CONNECT)
*                                      I - PROVIDES THREAD (ALL OTHERS)
Z901_RESULT    DS  F                   O - RECEIVES RESULT SET (GET)
*                                      I - PROVIDES RESULT SET (FETCH)
Z901_RECORDS   DS  F                   O - RECEIVES RECORD COUNT (GET)
Z901_CRITERIA_L DS F                   I - PROVIDES CRITERIA LENGTH
Z901_GET_L     DS  F                   O - RECEIVES GET LENGTH
*                                      I - PROVIDES GET LENGTH
Z901_CONTEXT   DS  CL8                 I - PROVIDES CONTEXT
Z901_SCOPE     DS  CL8                 I - PROVIDES SCOPE
Z901_CRITERIA  DS  CL160               I - PROVIDES CRITERIA
A100SR14 DS    F                       SAVE REGISTER 14
A200SR14 DS    F                       SAVE REGISTER 14
A300SR14 DS    F                       SAVE REGISTER 14
A400SR14 DS    F                       SAVE REGISTER 14
A500SR14 DS    F                       SAVE REGISTER 14
A900SR14 DS    F                       SAVE REGISTER 14
Z900SR14 DS    F                       SAVE REGISTER 14
         DS    0D                      ALIGN STORAGE
CM401ST  DS    CL(CM401COM_LENGTH)     STORAGE FOR CM401COM
```

```
         DS    ØD                      ALIGN STORAGE
CM4Ø2ST  DS    CL(CM4Ø2COM_LENGTH)     STORAGE FOR CM4Ø2COM
         DS    ØD                      ALIGN STORAGE
CM4Ø3ST  DS    CL(CM4Ø3COM_LENGTH)     STORAGE FOR CM4Ø3COM
         DS    ØD                      ALIGN STORAGE
CM431ST  DS    CL(CM431TSQ_LENGTH)     STORAGE FOR CM431TSQ
         DS    ØD                      ALIGN STORAGE
RECTRNC  DS    CL(TRANCLAS_TBL_LEN)    STORAGE FOR TRANCLAS
DYNSTORL EQU   *-DYNSTOR               LENGTH OF DYNAMIC STORAGE
*- CM431 Dynamic Storage - End                                        -*
         EJECT
*- Register Equates                                                   -*
         DFHREGS                       CICS STANDARD EQUATES
B1       EQU   3                       BASE CODE REGISTER
EIBREG   EQU   11                      EXEC INTERFACE BLOCK REGISTER
B2       EQU   12                      BASE CODE REGISTER
DYNREG   EQU   13                      DYNAMIC STORAGE REGISTER
         EJECT
*=       E  N  T  R  Y          P  O  I  N  T                         =*
CM431    DFHEIENT CODEREG=(B1,B2),DATAREG=(DYNREG),EIBREG=(EIBREG)
CM431    AMODE 31
CM431    RMODE ANY
*- Program Identification "Eye-Catchers"                              -*
         B     A000_MAINLINE           BRANCH OVER EYE-CATCHERS
ASMEYE   DC    C'*'                    ASTERISK
ASMPROG  DC    C'CM431   '             PROGRAM NAME
         DC    C'-'                    HYPHEN
ASMLVL   DC    C'CWM00001'             PROGRAM LEVEL
         DC    C' '                    BLANK
ASMDATE  DC    C'&SYSDATE'             DATE OF ASSEMBLY
         DC    C' '                    BLANK
ASMTIME  DC    C'&SYSTIME'             TIME OF ASSEMBLY
         DC    C'*'                    ASTERISK
ASMEYEL  EQU   *-ASMEYE                LENGTH OF EYE-CATCHER
         EJECT
*- A Ø Ø Ø _ M A I N L I N E :  Controls the flow of the program    -*
*- R2   DSECT - CM431COM                                             -*
*- R3   BASE CSECT CM431                                             -*
*- R8   DSECT - CM4Ø1COM                                             -*
*- R9   DSECT - CM4Ø2COM                                             -*
*- R1Ø  DSECT - CM4Ø3COM                                             -*
*- R11  EIBREG DSECT - DFHEIBLK                                      -*
*- R12  BASE CSECT CM431                                             -*
*- R13  DYNREG DSECT - DFHEISTG                                      -*
*- R14  Linkage                                                      -*
A000_MAINLINE  DS  ØH
         BAL   R14,A1Ø0_INITIALIZE     PERFORM INITIALIZATION
         CLC   RETC,=F'4'                IF RETC > 4
         BH    A000TERM                    THEN TERMINATE
         BAL   R14,A2Ø0_CPSM_CONN      CONNECT TO CPSM
```

```
         CLC   RETC,=F'4'                IF RETC > 4
         BH    A000TERM                    THEN TERMINATE
         BAL   R14,A300_CPSM_GET       GET OBJECT(TRANCLAS)
         CLC   RETC,=F'4'                IF RETC => 4
         BNL   A000DISC                    THEN DISC/TERM
         BAL   R14,A400_CPSM_FETCH     FETCH TRANCLAS RECORDS
A000DISC BAL   R14,A500_CPSM_DISC      DISCONNECT FROM CPSM
A000TERM BAL   R14,A900_TERMINATION    TERMINATION
*- A 0 0 0 _ R E T U R N _ T O _ C I C S                          -*
A000_RETURN_TO_CICS  DS  0H
RETURN   EXEC  CICS RETURN
*=       E  X  I  T            P  O  I  N  T                      =*
         EJECT
*- A 1 0 0 _ I N I T I A L I Z E :  Perform Initialization       -*
*- R2   DSECT - CM431COM                                         -*
*- R3   BASE CSECT CM431                                         -*
*- R4   Work Register                                            -*
*- R5   Work Register                                            -*
*- R6   Work Register                                            -*
*- R7   Work Register                                            -*
*- R8   DSECT - CM401COM                                         -*
*- R9   DSECT - CM402COM                                         -*
*- R10  DSECT - CM403COM                                         -*
*- R11  EIBREG DSECT - DFHEIBLK                                  -*
*- R12  BASE CSECT CM431                                         -*
*- R13  DYNREG DSECT - DFHEISTG                                  -*
*- R14  Linkage                                                  -*
A100_INITIALIZE  DS  0H
*- Clear DYNSTOR.                                                -*
         LA    R4,DYNSTOR              ADDRESS DYNSTOR
         LA    R5,DYNSTORL             LENGTH OF DYNSTOR
         XR    R6,R6                   FROM ADDRESS NOT REQUIRED
         XR    R7,R7                   SET LENGTH TO 0
         ICM   R7,8,=C' '              SET PADDING TO BLANKS
         MVCL  R4,R6                   CLEAR STORAGE TO BLANKS
         ST    R14,A100SR14            SAVE REGISTER 14
         XC    RETC,RETC               CLEAR RETURN CODE
*- Address CICS EIB                                              -*
EIBADD   EXEC  CICS ADDRESS                                       X
               EIB(EIBREG)
*- Establish addressability and map CM401COM, CM402COM and CM403COM  -*
         LA    R8,CM401ST              ADDRESS CM401COM
         USING CM401COM,R8             MAP CM401COM
         MVC   CM401COM_EYECATCH,=C'<<  CM401COM  >>'  EYECATCHER
         LA    R9,CM402ST              ADDRESS CM402COM
         USING CM402COM,R9             MAP CM402COM
         MVC   CM402COM_EYECATCH,=C'<<  CM402COM  >>'  EYECATCHER
         LA    R10,CM403ST             ADDRESS CM403COM
         USING CM403COM,R10            MAP CM403COM
         MVC   CM403COM_EYECATCH,=C'<<  CM403COM  >>'  EYECATCHER
```

```
*- Establish SYSID                                                    -*
A100ASGN EXEC  CICS ASSIGN                                             X
               SYSID(SYSID)                                            X
               RESP(CM402COM_RESP1)                                    X
               RESP2(CM402COM_RESP2)
         MVC   CM402COM_FUNC(L'CM402COM_FUNC),EIBFN   MOVE FUNCTION
         CLC   CM402COM_RESP1,DFHRESP(NORMAL)    NORMAL COMPLETION?
         BE    A100COMM                YES - ADDRESS COMMAREA
*                                      NO  - PROCESS ERROR
         BAL   R14,Z402_CICS_DIAGS     PERFORM CICS DIAGNOSTICS
         MVC   CM401COM_MSGNO,=CL(L'CM401COM_MSGNO)'CM431110E' NUMBER
         MVC   CM401COM_TEXT(L'CM431110E),CM431110E            TEXT
         BAL   R14,Z401_ISSUE_MESSAGE  ISSUE MESSAGE
         MVC   RETC,=F'110'            SET RETURN CODE
         B     A100DUMP                DUMP TRANSACTION
*- Check that a COMMAREA has been provided, if it isn't present     -*
*- issue an error message and return control. If a COMMAREA is      -*
*- present then map it using the DSECT CM431COM.                    -*
A100COMM DS    0H                      CHECK COMMAREA
         OC    EIBCALEN,EIBCALEN       ANY COMMAREA?
         BNZ   A100ACOM                YES - ADDRESS COMMAREA
*                                      NO  - BUILD MESSAGE
         MVC   CM401COM_MSGNO,=CL(L'CM401COM_MSGNO)'CM431113E' NUMBER
         MVC   CM401COM_TEXT(L'CM431113E),CM431113E            TEXT
         BAL   R14,Z401_ISSUE_MESSAGE  ISSUE ERROR MESSAGE
         MVC   RETC,=F'113'            SET RETURN CODE
A100DUMP MVC   DUMPCODE,=C'A100'       SET DUMP CODE
         BAL   R14,Z900_DUMP_TRAN      DUMP TRANSACTION
         B     A100RET                 RETURN
A100ACOM DS    0H                      ADDRESS AND MAP CM431 COMMAREA
         L     R2,DFHEICAP             ADDRESS COMMAREA
         USING CM431COM,R2             MAP COMMAREA
         MVC   CM431COM_EYECATCH,=C'<<  CM431COM  >>'  EYECATCHER
         XC    CM431COM_RC,CM431COM_RC INITIALIZE RC
         XC    Z901_RESULT,Z901_RESULT INITIALIZE RESULT
*- Return to caller                                                  -*
A100RET  L     R14,A100SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
*- A 2 0 0 _ C P S M _ C O N N  :  Connect to CPSM                    -*
*- R2   DSECT - CM431COM                                             -*
*- R3   BASE CSECT CM431                                             -*
*- R8   DSECT - CM401COM                                             -*
*- R9   DSECT - CM402COM                                             -*
*- R10  DSECT - CM403COM                                             -*
*- R11  EIBREG DSECT - DFHEIBLK                                      -*
*- R12  BASE CSECT CM431                                             -*
*- R13  DYNREG DSECT - DFHEISTG                                      -*
*- R14  Linkage                                                      -*
A200_CPSM_CONN  DS  0H
```

```
         ST    R14,A200SR14            SAVE REGISTER 14
*- Establish whether CONTEXT and SCOPE have been provided.          -*
         CLC   CM431COM_CONTEXT,=CL8' '  IF BLANK
         BE    A200CT01                   THEN SET DEFAULT
         MVC   Z901_CONTEXT,CM431COM_CONTEXT ELSE USE CONTEXT PROVIDED
         B     A200SC01                   AND CHECK SCOPE
A200CT01 MVC   Z901_CONTEXT(L'CICSPLEX),CICSPLEX  SET CICSPLEX PREFIX
         MVC   Z901_CONTEXT+1(L'Z901_CONTEXT-1),=CL7'PLEX' ADD 'PLEX'
         MVC   CM431COM_CONTEXT,Z901_CONTEXT  MOVE DEFAULT TO COMMAREA
A200SC01 CLC   CM431COM_SCOPE,=CL8' '    IF BLANK
         BE    A200SC02                   THEN SET DEFAULT
         MVC   Z901_SCOPE,CM431COM_SCOPE     ELSE USE SCOPE PROVIDED
         B     A200CONN                   AND CONNECT TO CPSM
A200SC02 MVC   Z901_SCOPE,Z901_CONTEXT   SET SCOPE = CONTEXT
         MVC   CM431COM_SCOPE,Z901_SCOPE      MOVE DEFAULT TO COMMAREA
*- CONNECT to CPSM                                                  -*
A200CONN EXEC  CPSM CONNECT                                          X
                    CONTEXT(Z901_CONTEXT)                            X
                    SCOPE(Z901_SCOPE)                                X
                    VERSION(=CL4'0140')                              X
                    THREAD(Z901_THREAD)                              X
                    RESPONSE(Z901_RESPONSE)                          X
                    REASON(Z901_REASON)
         CLC   Z901_RESPONSE,EYUVALUE(OK)     RESPONSE OK?
         BE    A200RET                YES - RETURN
*                                     NO  - PROCESS ERROR
         MVC   CM401COM_MSGNO,=CL(L'CM401COM_MSGNO)'CM431200E' NUMBER
         MVC   CM401COM_TEXT(L'CM431200E),CM431200E            TEXT
         BAL   R14,Z401_ISSUE_MESSAGE  ISSUE MESSAGE
         BAL   R14,Z901_CPSM_DIAGS     PERFORM CPSM DIAGNOSTICS
         MVC   RETC,=F'200'            SET RETURN CODE
A200DUMP MVC   DUMPCODE,=C'A200'       SET DUMP CODE
         BAL   R14,Z900_DUMP_TRAN      DUMP TRANSACTION
*- Return to caller                                                 -*
A200RET  L     R14,A200SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
*- A 3 0 0 _ C P S M _ G E T  :  GET CPSM OBJECT(TRANCLAS)           -*
*- R2   DSECT - CM431COM                                            -*
*- R3   BASE CSECT CM431                                            -*
*- R8   DSECT - CM401COM                                            -*
*- R9   DSECT - CM402COM                                            -*
*- R10  DSECT - CM403COM                                            -*
*- R11  EIBREG DSECT - DFHEIBLK                                     -*
*- R12  BASE CSECT CM431                                            -*
*- R13  DYNREG DSECT - DFHEISTG                                     -*
*- R14  Linkage                                                     -*
A300_CPSM_GET  DS  0H
         ST    R14,A300SR14            SAVE REGISTER 14
*- Establish whether CRITERIA provided.                             -*
```

```
         CLC   CM431COM_CRITERIA(1),=C' '  IF BLANK
         BE    A300CT01                        THEN SET DEFAULT
         MVC   Z901_CRITERIA,CM431COM_CRITERIA  ELSE USE CRITERIA
         MVC   Z901_CRITERIA_L,CM431COM_CRITERIA_L   AND LENGTH
         B     A300GET                           GET OBJECT
A300CT01 MVC   Z901_CRITERIA,=CL7'NAME=*.' DEFAULT CRITERIA
         MVC   Z901_CRITERIA_L,=F'7'       DEFAULT CRITERIA LENGTH
         MVC   CM431COM_CRITERIA,Z901_CRITERIA   DEFAULT IN COMMAREA
         MVC   CM431COM_CRITERIA_L,Z901_CRITERIA_L   DEFAULT IN COMM.
*- GET OBJECT(TRANCLAS)                                              -*
A300GET  EXEC  CPSM GET                                               X
                    OBJECT(=CL8'TRANCLAS')                            X
                    CRITERIA(Z901_CRITERIA)                           X
                    LENGTH(Z901_CRITERIA_L)                           X
                    COUNT(Z901_RECORDS)                               X
                    RESULT(Z901_RESULT)                               X
                    THREAD(Z901_THREAD)                               X
                    RESPONSE(Z901_RESPONSE)                           X
                    REASON(Z901_REASON)
         CLC   Z901_RESPONSE,EYUVALUE(OK)     RESPONSE OK?
         BE    A300SET                 YES - RETURN
*                                      NO  - CHECK FOR NODATA
         CLC   Z901_RESPONSE,EYUVALUE(NODATA) RESPONSE NODATA?
         BE    A300NODA                YES - SET CODES
*                                      NO  - PROCESS ERROR
         MVC   CM401COM_MSGNO,=CL(L'CM401COM_MSGNO)'CM431300E' NUMBER
         MVC   CM401COM_TEXT(L'CM431300E),CM431300E            TEXT
         BAL   R14,Z401_ISSUE_MESSAGE  ISSUE MESSAGE
         BAL   R14,Z901_CPSM_DIAGS     PERFORM CPSM DIAGNOSTICS
         MVC   RETC,=F'300'            SET RETURN CODE
A300DUMP MVC   DUMPCODE,=C'A300'        SET DUMP CODE
         BAL   R14,Z900_DUMP_TRAN      DUMP TRANSACTION
         B     A300RET                 RETURN
A300SET  MVC   CM431COM_RECORDS,Z901_RECORDS  SET RECORDS IN COMMAREA
         B     A300RET                 RETURN
A300NODA MVC   RETC,=F'4'              SET RETURN CODE
         MVC   CM431COM_RECORDS,=F'0'  SET NO. OF. RECS TO ZERO
*- Return to caller                                                  -*
A300RET  L     R14,A300SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
*- A 4 0 0 _ C P S M _ F E T C H  :  FETCH CPSM OBJECT(TRANCLAS)     -*
*- R2   DSECT - CM431COM                                             -*
*- R3   BASE CSECT CM431                                             -*
*- R4   Work Register - No. of Records & Loop Counter                -*
*- R5   Work Register - DSECT TRANCLAS                               -*
*- R6   Work Register - DSECT CM431TSQ                               -*
*- R8   DSECT - CM401COM                                             -*
*- R9   DSECT - CM402COM                                             -*
*- R10  DSECT - CM403COM                                             -*
```

```
*- R11  EIBREG DSECT - DFHEIBLK                                      -*
*- R12  BASE CSECT CM431                                             -*
*- R13  DYNREG DSECT - DFHEISTG                                      -*
*- R14  Linkage                                                      -*
A400_CPSM_FETCH  DS  0H
         ST    R14,A400SR14            SAVE REGISTER 14
*- Establish whether a temporary storage queue name has been         -*
*- provided, if not the data is not required.                        -*
         CLC   CM431COM_TSQ,=CL16' '       IF BLANK
         BE    A400RET                       THEN NO TSQ REQUIRED
*                                            ELSE CREATE TSQ
*- Loop through all the records and write them to the TSQ           -*
A400ADDR L     R4,Z901_RECORDS             NO. OF RECORDS/LOOP COUNTER
         LA    R5,RECTRNC                  -->  TRANCLAS STORAGE
         USING TRANCLAS,R5                 AND MAP IT
         LA    R6,CM431ST                  -->  CM431TSQ STORAGE
         USING CM431TSQ,R6                 AND MAP IT
A400LOOP DS    0H                          START OF LOOP
*- FETCH OBJECT(TRANCLAS)                                           -*
         MVC   Z901_GET_L,=A(TRANCLAS_TBL_LEN)  SET LENGTH FOR FETCH
A400FTCH EXEC  CPSM FETCH                                             X
                    INTO(TRANCLAS)                                    X
                    LENGTH(Z901_GET_L)                                X
                    RESULT(Z901_RESULT)                               X
                    THREAD(Z901_THREAD)                               X
                    RESPONSE(Z901_RESPONSE)                           X
                    REASON(Z901_REASON)
         CLC   Z901_RESPONSE,EYUVALUE(OK)     RESPONSE OK?
         BE    A400TSQ                 YES - ADD TO TSQ
*                                      NO  - PROCESS ERROR
         MVC   CM401COM_MSGNO,=CL(L'CM401COM_MSGNO)'CM431400E' NUMBER
         MVC   CM401COM_TEXT(L'CM431300E),CM431400E            TEXT
         BAL   R14,Z401_ISSUE_MESSAGE  ISSUE MESSAGE
         BAL   R14,Z901_CPSM_DIAGS     PERFORM CPSM DIAGNOSTICS
         MVC   RETC,=F'400'            SET RETURN CODE
A400DUMP MVC   DUMPCODE,=C'A400'        SET DUMP CODE
         BAL   R14,Z900_DUMP_TRAN      DUMP TRANSACTION
         B     A400RET                 RETURN
A400TSQ  MVC   CM431TSQ_REGION,TRANCLAS_EYU_CICSNAME       REGION NAME
         MVC   CM431TSQ_TRANCLAS,TRANCLAS_NAME             TRANCLAS
         MVC   CM431TSQ_MAXACTIVE,TRANCLAS_MAXACTIVE       MAXACTIVE
         MVC   CM431TSQ_ACTIVE,TRANCLAS_ACTIVE             ACTIVE
         MVC   CM431TSQ_QUEUED,TRANCLAS_QUEUED             QUEUED
         MVC   CM431TSQ_PURGETHRESH,TRANCLAS_PURGETHRESH   PURGETHRESH
A400TSQW EXEC  CICS WRITEQ TS                                         X
               QNAME(CM431COM_TSQ)                                    X
               FROM(CM431TSQ)                                         X
               LENGTH(=AL2(CM431TSQ_LENGTH))                          X
               RESP(CM402COM_RESP1)                                   X
               RESP2(CM402COM_RESP2)
```

```
         MVC   CM402COM_RESOURCE,EIBRSRCE             MOVE RESOURCE
         MVC   CM402COM_FUNC(L'CM402COM_FUNC),EIBFN   MOVE FUNCTION
         CLC   CM402COM_RESP1,DFHRESP(NORMAL)    NORMAL COMPLETION?
         BE    A400LPBK                YES - LOOP BACK FOR NEXT RECORD
*                                      NO  - PROCESS ERROR
         BAL   R14,Z402_CICS_DIAGS     PERFORM CICS DIAGNOSTICS
         MVC   CM401COM_MSGNO,=CL(L'CM401COM_MSGNO)'CM431410E' NUMBER
         MVC   CM401COM_TEXT(L'CM431410E),CM431410E            TEXT
         BAL   R14,Z401_ISSUE_MESSAGE  ISSUE MESSAGE
         MVC   RETC,=F'410'            SET RETURN CODE
         B     A400DUMP                DUMP TRANSACTION
A400LPBK BCT   R4,A400LOOP             LOOP BACK/END OF LOOP
         DROP  R5                      DROP TRANCLAS
         DROP  R6                      DROP CM431TSQ
*- Return to caller                                              -*
A400RET  L     R14,A400SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
*- A 5 0 0 _ C P S M _ D I S C  :  Disconnect from CPSM           -*
*- R2   DSECT - CM431COM                                          -*
*- R3   BASE CSECT CM431                                          -*
*- R8   DSECT - CM401COM                                          -*
*- R9   DSECT - CM402COM                                          -*
*- R10  DSECT - CM403COM                                          -*
*- R11  EIBREG DSECT - DFHEIBLK                                   -*
*- R12  BASE CSECT CM431                                          -*
*- R13  DYNREG DSECT - DFHEISTG                                   -*
*- R14  Linkage                                                   -*
A500_CPSM_DISC  DS  0H
         ST    R14,A500SR14            SAVE REGISTER 14
*- TERMINATE CPSM API                                            -*
A500TERM EXEC  CPSM TERMINATE                                     X
                    RESPONSE(Z901_RESPONSE)                       X
                    REASON(Z901_REASON)
         CLC   Z901_RESPONSE,EYUVALUE(OK)     RESPONSE OK?
         BE    A500RET                 YES - RETURN
*                                      NO  - PROCESS ERROR
         MVC   CM401COM_MSGNO,=CL(L'CM401COM_MSGNO)'CM431500E' NUMBER
         MVC   CM401COM_TEXT(L'CM431500E),CM431500E            TEXT
         BAL   R14,Z401_ISSUE_MESSAGE  ISSUE MESSAGE
         BAL   R14,Z901_CPSM_DIAGS     PERFORM CPSM DIAGNOSTICS
         MVC   RETC,=F'500'            SET RETURN CODE
A500DUMP MVC   DUMPCODE,=C'A500'       SET DUMP CODE
         BAL   R14,Z900_DUMP_TRAN      DUMP TRANSACTION
*- Return to caller                                              -*
A500RET  L     R14,A500SR14            RESTORE REGISTER 14
         BR    R14                     RETURN TO CALLER
         EJECT
```

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

*Carl Wade McBurnie*
*IT Consultant (Germany)*

HostBridge Technology has announced HostBridge Version 4.0. This new version includes the latest release of its software for XML-enabling CICS applications, along with a new set of functionality called HostBridge Extended.

HostBridge Extended (HBX) includes four new features, first among them is a new CICS-native process automation engine. Using the process automation engine in HostBridge Extended, non-mainframe developers can access data, run transactions, and execute CICS commands using a single HTTP, SOAP, or MQ request.

The HostBridge process automation engine allows system architects, process designers, and application developers to write CICS processes using JavaScript. HostBridge Extended includes a workstation-based Interactive Development Environment (IDE) that makes it easy to develop and test process automation scripts for CICS. The resulting scripts are compiled into native CICS.

For further information contact:
HostBridge Technology, 1414 S Sangre Rd, Stillwater, OK 74074, USA.
Tel: (866) 965 2427.
URL: http://www.hostbridge.com/products/extended.

* * *

Rosebud Management Systems has announced Version 3.13 of Eden Server. Eden, which is a fully-featured COBOL/CICS emulator, is claimed to reduce the costs associated with mainframe-based legacy applications by porting them to Windows.

Eden Client provides full support for CICS/BMS functionality in a Windows GUI without the need for changes, even to programs utilizing low-level BMS features. Eden Client is provided in two configurations; one follows a thick distributed architecture and the second uses a server-based client with a thin workstation interface suitable for use over the Internet or a corporate extranet. Eden Server and Client both offer a range of programming APIs that enable customers to integrate Eden with a variety of systems written in other languages, including systems hosted on other platforms.

For further information contact:
Rosebud Management Systems, 216 Pleasant Hill Road, Flanders, NJ 07836, USA.
Tel: (866) 767 3283.
URL: http://www.rosebudusa.com/products.html.

* * *

IBM is modernizing COBOL applications by bridging its mainframe-oriented COBOL and WebSphere products to EJB and Service-Oriented Architectures (SOA) in new versions of its Enterprise COBOL and WebSphere Studio Enterprise developer products.

Version 3.3 adds the ability to generate XML and hand it to other applications. Developers can write an EJB in COBOL 3.3 on WebSphere z/OS. This allows developers to create reusable components and use those components in different environments, such as in a batch application or in a CICS application.

Also in the new release, WebSphere applications can communicate with CICS applications without users having to buy WebSphere Studio Application Developer Enterprise Integrator separately.

For further information contact your local IBM representative.