223

CICS update

June 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.

CICSPlex SM dynamic workload management

CICSPLEX SM EXIT CUSTOMIZATION

In this article we will look at the exit customization capabilities provided by CICSPlex SM when routing various types of CICS workload. Why should you do so? Only you know your transactions!

CICSPlex SM provides a routing exit (EYU9WRAM) to its CICS exit (EYU9XLOP), which provides the ability to customize workload routing. Through interrogation of the TIOA, COMMAREA, modification of TIOA data via EXEC CICS RETURN INPUTMSG, or direct modification of the COMMAREA, you can enhance the workload management of your applications. Furthermore, we have up to now used affinity definitions to control when affinities are created and destroyed. Examination of the above control blocks, shows that exit customization allows for the dynamic creation and destruction of affinity elements via explicit calls to CICSPlex SM.

The dynamic routing API (DTRAPI) provides:

- SCOPE – to determine the candidate set of targets.
- BALANCE – to choose the best target.
- ROUTE – to request specific routing to the target.
- CREAFF – dynamically to create an affinity.
- DELAFF – dynamically to delete an affinity.

In order to understand the various calls, one must understand the routing models of CICS. These are outlined below. For a full description, see *CICS and the CICSPlex SM dynamic workload management model*, *CICS Update*, December 2002 (Issue 205). The parameters provided by CICS via the DFHDYPDS control block on various routing calls are detailed below.

DTRPGM MODEL

Figure 1 illustrates the DTRPGM model.

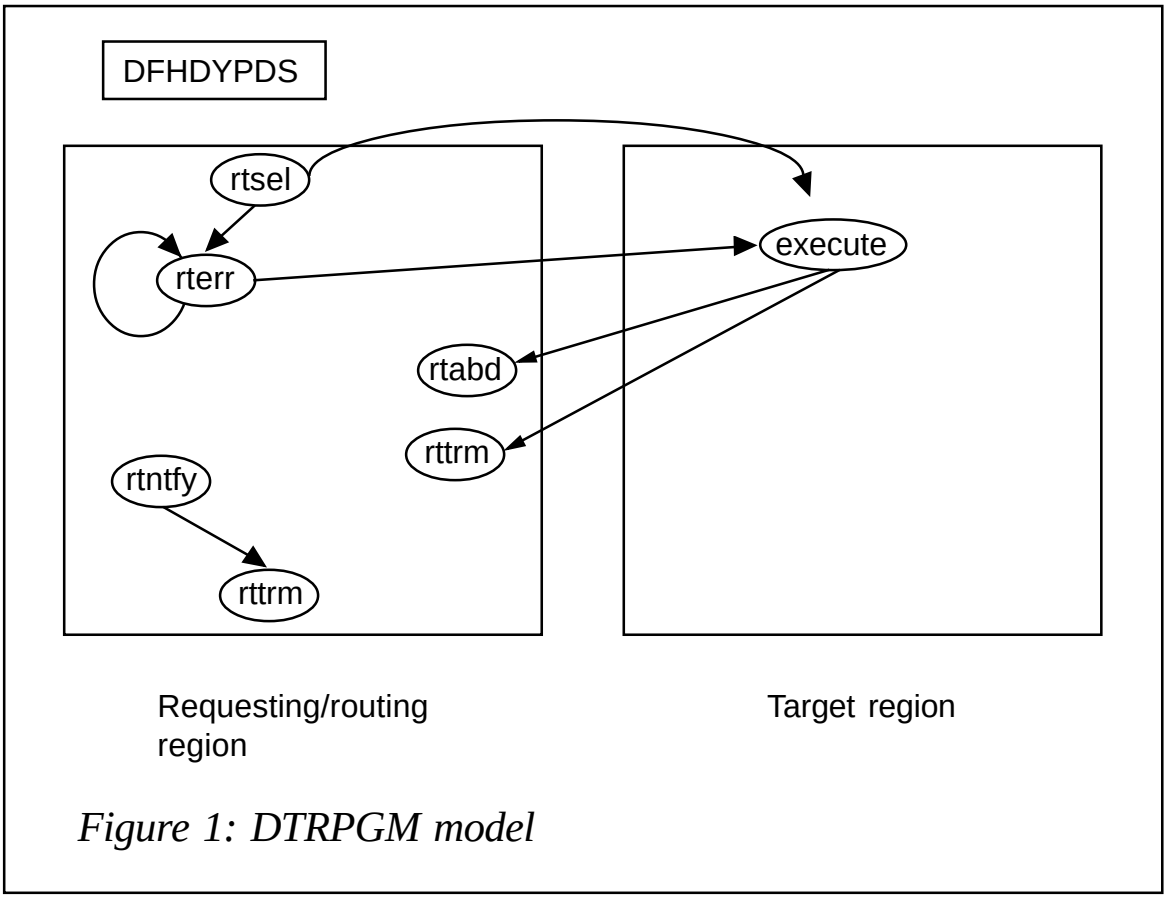

Figure 1: DTRPGM model

Common fields

There are many fields common to all RT-type routing calls. These are principally to communicate routing error codes, queueing options, response codes back to CICS, transaction priorities, abend control options, and MVS WLM workload service tokens. Access is also provided to the program's COMMAREA and TIOA. A user area is also provided for storing state data between calls.

Static routing

Identification of the static routing call type is provided via DYRTYPE and DYRFUNC. The exit is told the target region via DYRSYSID and DYRNETNM.

Dynamic transaction routing

Dynamic transaction routing is similar in form to static routing except that the exit can now choose the target and return it to CICS via DYRSYSID or DYRNETNM. The name of a program to be invoked if routing is not successful is also specifiable (DYRLPROG).

Start termid and start termid data

Start termid and start termid data have a form similar to the examples above.

Dynamic DPL

Dynamic DPL again has a similar form. However, the tranid has been deduced by CICS via explicit naming in the incoming Web URL, RDO PROGRAM definition, or, by default, CSMI.

Dynamic bridge

Dynamic bridge again has a similar form. In addition, the bridge facility token is provided by DYRBRTK.

DSRTPGM REQUESTS

Figure 2 illustrates DSRTPGM requests.

Common fields

Although different from RT routing, DSRTPGM routing again has a common base set of parameters passed.

CBTS

CBTS identifies the routing type and targets as in RT cases, but in addition identifies a process and activities.

Dynamic START requests (non-termid related)

Dynamic START requests (non-termid related) are similar to RT-like routing in terms of data and route selection.

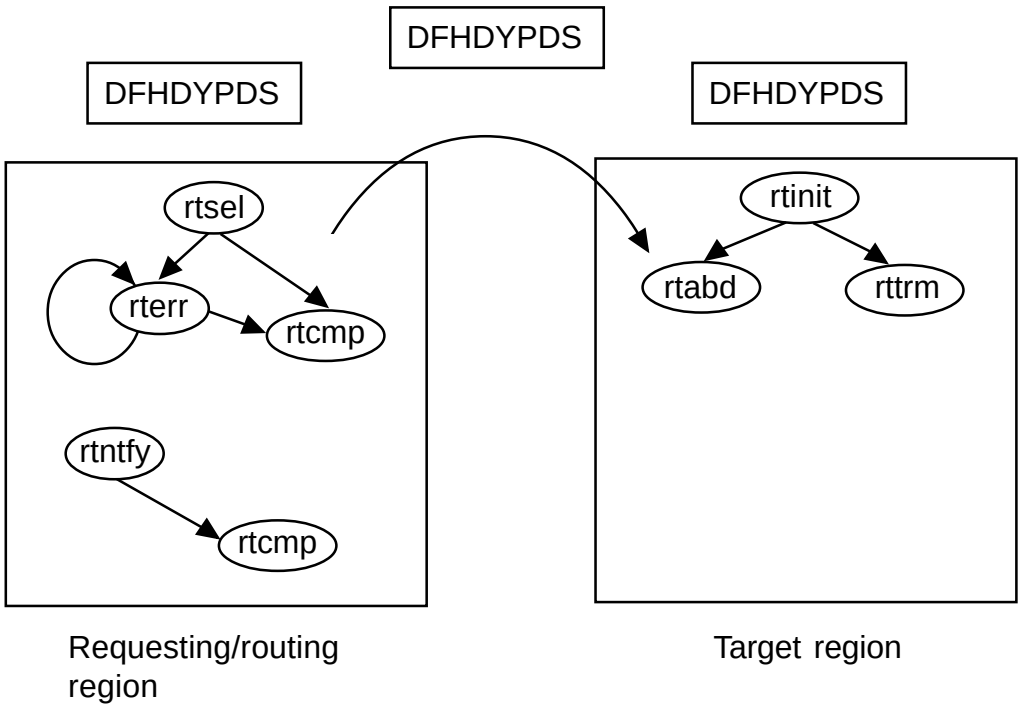

Figure 2: DSRTPGM requests

Dynamic IIOP requests

Dynamic IIOP requests are similar to those above.

CICS/CPSM ROUTING FLOWS

Figure 3 outlines the flow of control between CICS, CICSPlex SM routing, and the customizable WRAM. Several control blocks are passed, which are described below.

EYURWCOM

EYURWCOM is essentially the same as DFHDYPDS. However, queue/noqueue is not supported, since CICSPlex SM uses this as part of its controls. DYRLPROG and DYRABCDE are also not supported.

This control block also contains user-related data, DTRAPI communications fields, RTA event name, SCOPE/BALANCE

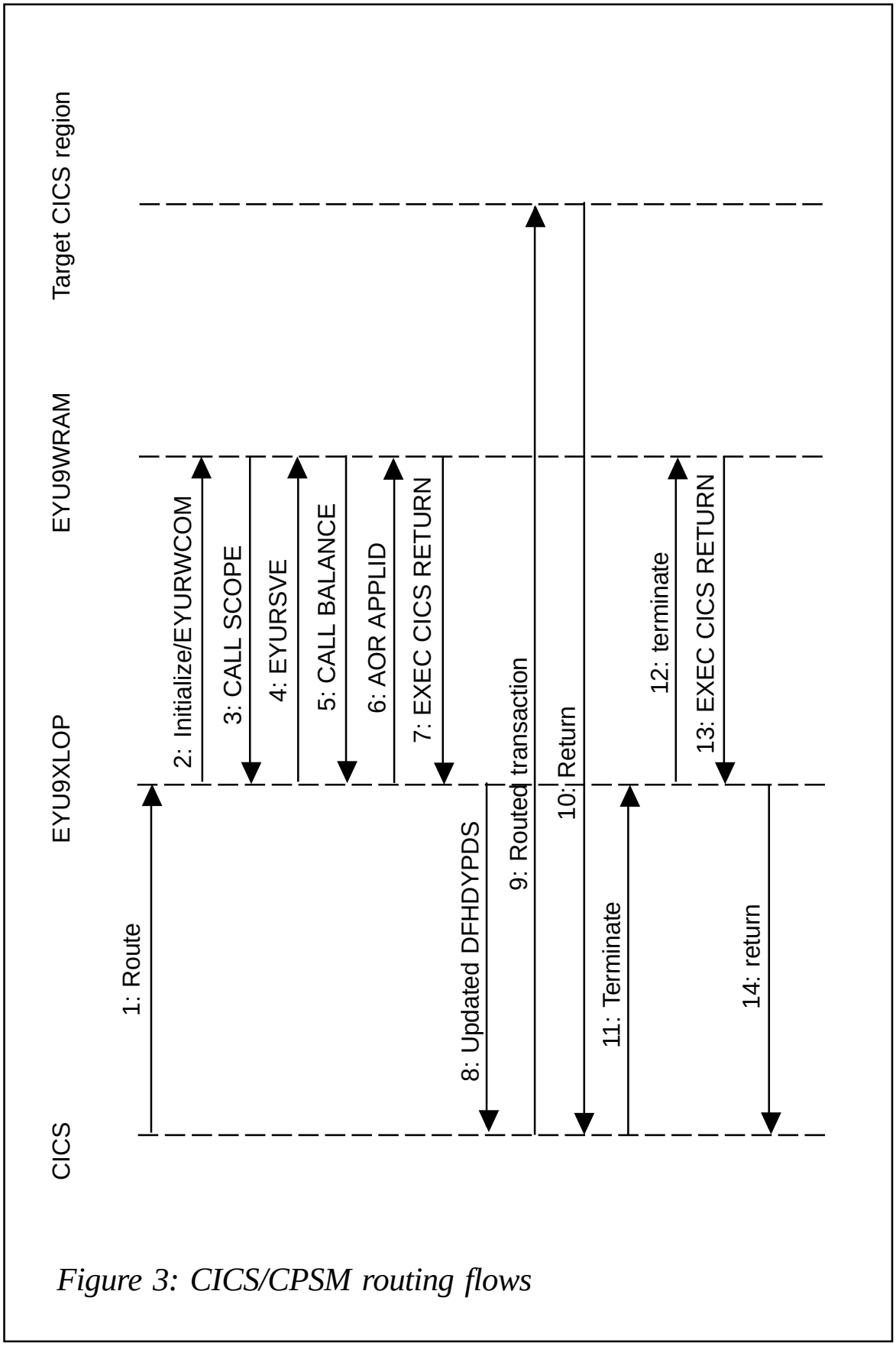

Figure 3: CICS/CPSM routing flows

availability, SCOPE vector list (mapped by EYURWSVE), and route error data.

EYURWSVE

There is one EYURWSVE per candidate target. It contains the CICS sysid, applid, connection status, target status, RTA eventlevel, etc. You can mark the target as ineligible for routing. The array of candidate targets is in order of eligibility.

ROUTING OUTSIDE THE ROUTING EXIT

In addition to being invoked by CICS, the routing code can be accessed directly by a user application via:

```
EXEC CICS LINK PROGRAM(EYU9XLOP) COMAREA(EYURWTRA)
```

This interface was initially provided for routing types of work not catered for by native CICS facilities. As more and more types of workload are catered for natively, the value of this interface has diminished. It is now available primarily for coexistence purposes.

Now you get to play with CICS. You must provide all the information that CICS would provide on a routing call. This includes all DFHDYPDS routing data, simulated transaction information, and workload classification data.

The selected sysid, applid and API responses are returned.

There is no support for signoff/logoff affinities (since the exits would never be invoked). It is only possible to use the queue-based algorithm in this way because you need to provide the service class token to the dynamic routing exit via EYURWTRA. Since there can be only one connection to MVS WLM per address space, and CICS has that connection, there is no way to obtain the srctoken.

Note that for DSRTPGM routing, the WRAM is called only in the routing region, not the target region (ie for init, term, abend). The flow is essentially the same.

Figure 4 illustrates routing outside the routing exit.

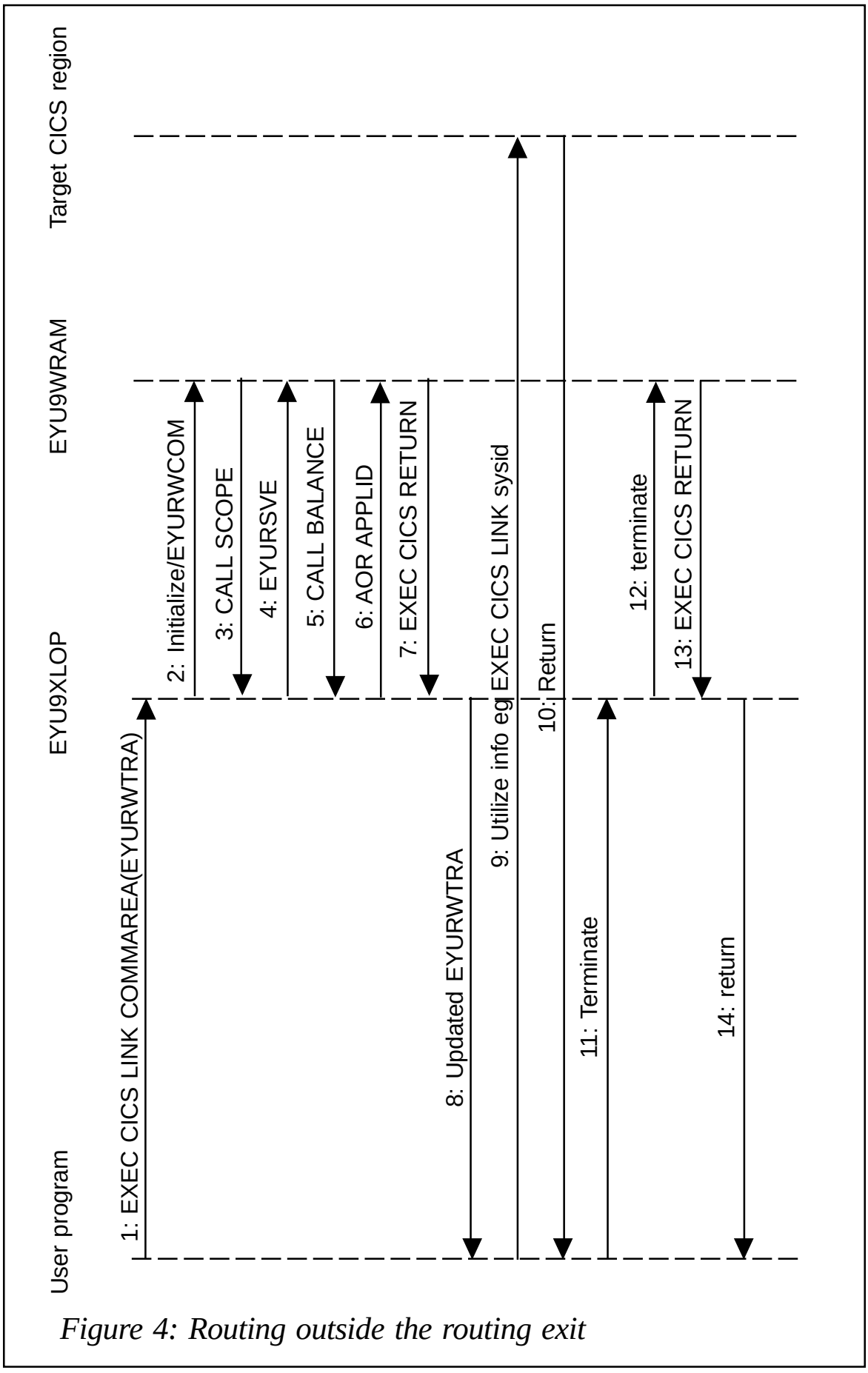

Figure 4: Routing outside the routing exit

Front-ending the routing exit

Finally there is a technique sometimes employed to modify the data provided by CICS via DFHDYPDS by front-ending EYU9XLOP. In this case the user creates his own routing exit (which he specifies to CICS). On invocation, it modifies the required fields and then invokes EYU9XLOP. This technique, though not officially supported, is utilized by some vendor packages as a means of primarily modifying the CICS tranid.

AN EXAMPLE OF USE

Scenario: a large bureau site is providing services to many clients that use common applications, but not necessarily at the same service/release level. Because of client restrictions, one level of the application cannot be simultaneously available to all users; each client moves to the next level of service following their own prescribed schedules.

When a client signs on, the location is saved in the TCTUA. When a routing request is made and EYU9WRAM is invoked, the client location is used to map to a pattern mask of valid AORs for this client. The mask is then applied to the sysids in the scope elements to restrict further the candidate targets.

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

A technique for writing a client/server application

INTRODUCTION

It is quite common to encounter client/server applications with partner programs executing on different platforms.

In most of these cases, the server is a CICS program, which

handles the data itself, while the client program is written and executed on a PC platform, and provides an 'attractive' graphical interface to an end user.

The communication between these two programs could be provided in one of several ways, via different network protocols and other auxiliary programs and tools.

PROBLEM ADDRESSED

The problem of consistency of the data flow between the client and the server program is of crucial importance.

Essentially the data flow is the 'traffic' of strings between the client and the server, and consistency is an 'awareness' by each of them that the other has received the string and 'done' its part. This consistency depends on the communication protocol used, and there is always a degree of uncertainty introduced by the media the data will go through.

However, regardless of the communication protocol applied, there is a way of writing a CICS server program and storing the data in it that, in correspondence with an appropriately written program (application) on a client side, provides full consistency of the data.

Here is a typical client/server data flow:

- The client program sends a string of data to a CICS program.
- The CICS program executes and sends an answer string back to the client.
- The client receives it and (optionally) does some additional work.

This is a typical example of an application that has several critical points when affected by a communication breakdown or unexpected end of any of the programs.

CLIENT SIDE CONSIDERATIONS

Let us consider all the possibilities that might occur in a client/

Case	*Client program*	*Communication*	*CICS program*	*Communication*	*Client program*
1	Failed	-	-	-	-
2	Sent data	Failed	-	-	-
3	Sent data	Passed data	Failed	-	-
4	Sent data	Passed data	Sent back answer	Failed	-
5	Sent data	Passed data	Sent back answer	Passed data	Failed
6	Sent data	Passed data	Sent back answer	Passed data	Received answer

Figure 1: Client side considerations

server data flow. Figure 1 shows us the possibilities.

In order to demonstrate a solution to possible data flow problems, we will assume certain preconditions that a client program has to satisfy.

A string sent from a client to a CICS program will have this structure (written as a PL/I structure variable):

```
1 client_string,
   2 transaction_name CHAR(4),
   2 do_cancel_flag   CHAR(1),
   2 client_data      CHAR(length of client_data);
```

The *do_cancel_flag* field can have two values – ‘1’ (do) and ‘0’ (cancel). Also, immediately before initializing the CICS program, the client program stores the *client_string* into a local *sent_strings_file* with this structure:

```
1 sent_strings_record,
   2 answer_received        CHAR(1) INIT('0'),
   2 client_string          CHAR(length of client_string),
   2 datetime_of_receiving  PIC'(17)9' INIT(0),
   2 answer_string          CHAR(length of answer_string) INIT('');
```

The *answer_received* field can have two values – ‘1’ (answer

received) and ‘0’ (answer not received). Its initial value is ‘0’.

Updating the *answer_received* to ‘1’, along with updating *datetime_of_receiving* and *answer_string*, must be the last action performed by a regularly-ended client/server application.

It is assumed that theclient program’s *current_datetime* format is adjusted to a PL/I DATETIME function formatted as year/month/day/hour/minute/second/microsecond.

Looking at the list of possible cases in Figure 1, we can conduct our analysis.

First case: client program has failed. It is important to notice that it does not matter what the reason was for its failure. It might not have been caused by a client’s application at all; it could have been an operating system error, power cut, etc. From an end user point of view, it is all the same – there is no answer. There are two possibilities – *sent_strings_record* either has or has not been written into a local *sent_strings_file*. As mentioned above, the client program writes *client_string* into a local *sent_strings_file* immediately before initializing the CICS program. Let’s suppose it has been done, and that the *answer_received* field = ‘0’. Once the client program recovers, regardless of the reason that caused its failure, it checks the *sent_strings_file* for a *sent_strings_record* with *answer_received* field = ‘0’. At this point, the client program does not know (and actually does not need to know) why it did not get an answer from the CICS program – in this particular case because it has not even sent the data. Now the client program checks the *do_cancel_flag* field value. If *do_cancel_flag*=‘1’ (do) it changes it into ‘0’ (cancel) and sends it again to CICS. And if the *do_cancel_flag*=‘0’ (cancel) it does the same without any changes. In this way, the client achieves consistency of the data, because it annuls any changes to the data that might have been made by a CICS program every time it does not get an answer from the CICS program.

Analysing the second, third, and fourth case in the table, we come to the conclusion that from the client’s perspective they are all the same, and like the first one, except that the client program

has not failed but simply did not get the answer within the certain predefined period of time (timeout). In third case, when the CICS program fails, it is very useful to limit the amount of time it's allowed to execute to a reasonable amount, so after a while it terminates its execution. It prevents possible problems for CICS, and could be done at a system level. That, of course, depends on the nature of the whole client/server application. It is very important to note that the client application must be able to recognize this timeout situation, and, if necessary, present an appropriate message to the end user. At this point, as opposed to the first case, the client program at least 'knows' one thing – it did not get any answer. What it does not know is whether the CICS program ended normally or failed, or even whether it started at all. It must afterwards (again depending on the nature of the whole application) immediately, or within a certain period of time, start the program again automatically with the same *client_string* and the *do_cancel_flag* set to '0'. By doing this, as in the first case, the consistency of the data is ensured.

Analysing the fifth case, we can conclude that this case practically turns into the first one. Everything is the same except the client program did not manage to end normally. Since the last action in the client program must be rewriting the *answer_received* from '0' to '1', the assumption is that its value is '0' (otherwise it has ended normally). This is the same as in the first case when the *sent_strings_record* has been written into a local *sent_strings_file*.

Eventually, after as many tries as necessary, the client program will (hopefully) get an answer from CICS, update the *answer_received* field to a value of '1', and add the *time_of_receiving* and *answer_string* field to a *sent_strings_record*. This is the sixth case – a normal end of the client/server application. The *sent_strings_record* of the successfully ended programs should be, occasionally, deleted from a local *sent_strings_file* and rewritten to another log file so as to reduce its size, leaving only unsuccessfully ended ones in it, as long as they exist.

CICS CODE CONSIDERATIONS

All the considerations given above direct us to the following: a CICS server program must be written in such a way that it performs one type of action with the *do_cancel_flag* value = '1', and another, closely tied to the first, with the *do_cancel_flag* value = '0'.

Furthermore, since the *client_string* has the same value in both cases except for the *do_cancel_flag*. The CICS program must be able to match the *client_data* received, with the same string that has been previously received and processed in the program. Such a request could be met by suitable programming and storing the data in the CICS program.

At this point it is very important to emphasize one thing: every client program should have an option available to an end user to choose to cancel some action that has been performed. So every CICS server program should contain this 'do_cancel' technique.

Accordingly, the CICS server program named PROGRAM1, linked to a transaction TRN1, should have a general structure similar to this one:

```
PROGRAM1: PROC OPTIONS(MAIN);
  -- /*****************************************/
  -- /* Declarations of the program variables */
  -- /*****************************************/
  DCL 1 received_client_string,
        2 transaction_name CHAR(4),  /* 'TRN1' */
        2 do_cancel_flag   CHAR(1),
        2 client_data      CHAR(length of client_data);
  DCL answer_string CHAR(length of answer_string) INIT('');
  /********************** program ************************/
  EXEC CICS SYNCPOINT;
  CALL RECEIVE_CLIENT_STRING;

  IF received_client_string.do_cancel_flag='0' THEN CALL
CANCEL_PREVIOUS_RECORD;
  CALL UPDATE_FILES;

  CALL SEND_ANSWER_STRING;

END PROGRAM1;
```

The code presented above is only a skeleton.

Here is one practical example – a simple financial transaction.

A client program sends an account number and an amount to debit or credit for a particular account. It gets an answer from CICS containing a new account balance and a transaction identification number, and (optionally) prints a document with an account number, amount, new account balance, and unique identification number of the transaction.

This transaction as a client/server application requires certain organization in storing data on either side. On the client side, everything is already defined with a *sent_strings_record* and a local *sent_strings_file*.

In the CICS program, according to the *client_data* and application's logic, a *transaction_file* is needed to store a record of each transaction in a *transaction_record*. Its structure must be part of the required declaration of the program variables. In addition an *account_file* is necessary. The *transaction_file* and the *account_file* could be VSAM files or DB2 tables. Both cases will be covered in the code of the procedures.

```
 /************************************************/
 /* necessary declaration of the program variables */
 /************************************************/
 /* structure of the client data */
 DCL 1 received_client_data BASED(ADDR(client_data)),
       2 account_number         PIC'(length of account_number)9',
       2 datetime_of_sending    PIC'(17)9',

/*********************************************************************/
/* Provide a uniqueness for client_data. Of course it could be       */
/* done in many different ways, via a unique transaction number on   */
/* client side or other. This one is taken to simplify the matter.   */
/*********************************************************************/

       2 amount                 PIC'(length of amount)9',
       2 debit_credit_flag      CHAR; /* - debit, + credit */

 /* structure of the transaction_record of a transaction_file */
 DCL 1 transaction_record,
       2 account_number             PIC'(length of account_number)9',
       2 datetime_of_sending        PIC'(17)9',
```

```
        2 unique_transaction_number  PIC'(15)9',

    /*****************************************************************/
    /* Current_date (PIC'(8)9') + transaction counter (PIC'(7)9')    */
    /* within a date.  It is not convenient to take current_datetime */
    /* as a unique_transaction_number because it might be smaller or */
    /* equal to the datetime_of_sending due to discrepancy between   */
    /* the times on different platforms. Of course, as has been      */
    /* said before datetime_of_sending is the simplest option in this */
    /* example, and a consequence is the structure of this number.   */
    /*****************************************************************/

        2 amount                      PIC'(length of amount)9',
        2 debit_credit_flag           CHAR,
        2 current_account_balance     PIC'(length of account_balance)9',
        2 do_cancel_flag              CHAR(1);

  DCL transaction_record_key CHAR(length of account_number + 32)
BASED(ADDR(transaction_record));
  DCL generic_transaction_record_key CHAR(length of account_number + 17)
BASED(ADDR(transaction_record));

   /*****************************************************************/
  /*Though unique_transaction_number itself is unique, unique key to */
  /* transaction_file consists of an account_number + unique field   */
  /*of a client_data + unique field of transaction_record on a server.*/
  /*This is generally a good approach of storing data in a CICS server*/
  /* program. A generic key of transaction_file is account_number +   */
/* datetime_of_sending and it would also be unique if it were not for */
/* do_cancel records with the equal values of the datetime_of_sending.*/
/*****************************************************************/

  DCL 1 max_unique_transaction_number
BASED(ADDR(unique_transaction_number)),
        2 date                  PIC'(8)9',
        2 transaction_counter   PIC'(7)9';

   /*****************************************************************/
  /* This field helps to calculate a unique_transaction_number   */
   /*****************************************************************/

  DCL current_datetime CHAR(17);
  DCL 1 redef_current_datetime BASED(ADDR(current_datetime)),
        2 current_date PIC'(8)9',
        2 rest         PIC'(9)9';
  /* Simplest possible structure of an account_file's record */
  DCL 1 account_record,
        2 account_number   PIC'(length of account_number)9',
        2 account_balance  PIC'(length of account_balance)9';
```

```
  DCL account_record_key CHAR(length of account_number)
BASED(ADDR(account_record));
  /* Account_number is an unique key to the account_file */
  DCL (DATETIME, SUBSTR, ADDR, CSTG, VERIFY) BUILTIN;
  DCL response_value BIN FIXED(31);
  DCL indmax BIN FIXED(15);
  DCL number_of_duplicate BIN FIXED(15) INIT(Ø);
  /* Case of DB2 tables */
  EXEC SQL DECLARE transaction_file TABLE (transaction_record);
  EXEC SQL DECLARE account_file TABLE (account_record);
  /*column's names and formats in DB2 tables are identical to the  */
  /* record's description and their unique keys are the same as in */
  /* the VSAM files   */
  EXEC SQL INCLUDE SQLCA;
  EXEC SQL INCLUDE SQLDA;

  /****************** procedures **********************/
  RECEIVE_CLIENT_STRING: PROC;

   /*****************************************************************/
   /* Receives string sent from a client. It is very useful to invoke*/
   /* some additional controls in such a procedure. In this case     */
   /* beside the validity checking of a received_client_string       */
   /* (though already done on a client side), a possibility of       */
 /*getting the duplicate string from a client with do_cancel_flag='1'*/
  /*(do), due to some unexpected error, could be a very annoying one.*/
   /*****************************************************************/

    current_datetime=DATETIME;

    /* General format of a RECEIVE command. */
    /* Depending on type of communication   */
    /* (APPC, hllapi ..) it has slight differences. */
    EXEC CICS RECEIVE INTO(received_client_string)
         LENGTH(length of received_client_string) RESP
(response_value);
    IF response_value¬=Ø THEN CALL ERROR_MESSAGE('Receive',
response_value);

    IF (received_client_string.do_cancel_flag¬='1'&
        received_client_string.do_cancel_flag¬='Ø') !
       (VERIFY(received_client_data.account_number,'Ø123456789')¬=Ø) !

(VERIFY(received_client_data.datetime_of_sending,'Ø123456789')¬=Ø)  !
       (VERIFY(received_client_data.ammount,'Ø123456789')¬=Ø)  !
       (received_client_data.debit_credit_flag¬='+' &
      received_client_data.debit_credit_flag¬='-') THEN
    DO;    /* Basic received_client_string checking */
      answer_string='Error in application. Wrong data received';
```

```
      CALL SEND_ANSWER_STRING;
    END;

    /* Checking received_client_data.account_number */
    /* and possible duplicate string */

    /* Case of DB2 table */
    EXEC SQL SELECT account_number
             INTO :account_record.account_number
             FROM account_file
             WHERE account_number=:received_client_data.account_number;
    SELECT(SQLCODE);
      WHEN(Ø);
      WHEN(1ØØ) DO;
         answer_string='Unknown account number
'!!received_client_data.account_number;
        CALL SEND_ANSWER_STRING;
      END;
      OTHER CALL ERROR_MESSAGE('Select account number', SQLCODE);
    END;

    IF received_client_string.do_cancel_flag='1' THEN
    DO;
      EXEC SQL SELECT COUNT(*) INTO :number_of_duplicate :indmax
           FROM transaction_file
           WHERE account_number=:received_client_data.account_number
AND

datetime_of_sending=:received_client_data.datetime_of_sending  AND
                do_cancel_flag='1';
      IF SQLCODE¬=Ø THEN CALL ERROR_MESSAGE('Select count', SQLCODE);
      IF number_of_duplicate>Ø THEN
      DO;
        answer_string='Error in application. Duplicate data received';
        CALL SEND_ANSWER_STRING;
      END;
    END;

    /* Case of VSAM file */
    account_record.account_number= received_client_data.account_number;
    EXEC CICS READ FILE(account_file) INTO(account_record)
RIDFLD(account_record_key)  RESPONSE(response_value);
    SELECT(response_value);
      WHEN(Ø);
      WHEN(DFHRESP(NOTFND)) DO;
        answer_string='Unknown account number
'!!received_client_data.account_number;
        CALL SEND_ANSWER_STRING;
      END;
```

```
      OTHER CALL ERROR_MESSAGE('Read', response_value);
    END;
    IF received_client_string.do_cancel_flag='1' THEN
    DO;

transaction_record.account_number=received_client_data.account_number;

transaction_record.datetime_of_sending=received_client_data.datetime_of_sending;
      EXEC CICS STARTBR FILE(transaction_file)
RIDFLD(generic_transaction_file_key) EQUAL
         GENERIC KEYLENGTH(length of account_number + 17)
RESPONSE(response_value);
      SELECT(response_value);
        WHEN(Ø) DO;
          EXEC CICS READNEXT FILE(transaction_file)
INTO(transaction_record)
                    RIDFLD(transaction_record_key)
RESPONSE(response_value);
          IF response_value¬=Ø THEN CALL ERROR_MESSAGE ('Readnext',
response_value);
          loop:
          DO WHILE(response_value=Ø &

transaction_record.account_number=received_client_data.account_number  &

transaction_record.datetime_of_sending=received_client_data.datetime_of_sending);
            IF transaction_record.do_cancel_flag='1' THEN
            DO;
          answer_string='Error in application. Duplicate data received';
              CALL SEND_ANSWER_STRING;
            END;
      EXEC CICS READNEXT FILE(transaction_file) INTO(transaction_record)
                RIDFLD(transaction_record_key) RESPONSE(response_value);
          END;
        END;
        WHEN(DFHRESP(NOTFND));  /* there is no duplicate record */
        OTHER CALL ERROR_MESSAGE('Startbr', response_value);
      END;
      EXEC CICS ENDBR FILE(transaction_file) RESPONSE(response_value);
 IF response_value¬=Ø THEN CALL ERROR_MESSAGE ('Endbr', response_value);
    END;
  END RECEIVE_CLIENT_STRING;

  CANCEL_PREVIOUS_RECORD: PROC;
    /*********************************************************/
    /* Searches for the previous record that matches         */
    /* received_client_data. If it was not found, since      */
    /* nothing happened, appropriate message goes to client. */
    /* Elsewhere previous record is 'cancelled'.             */
```

```
     /*********************************************************/
     /* Case of a DB2 tables                                  */
     /*********************************************************/
     EXEC SQL UPDATE transaction_file
              SET do_cancel='Ø'
              WHERE do_cancel='1' AND
account_number=:received_client_data.account_number
                    AND
datetime_of_sending=:received_client_data.datetime_of_sending;
     SELECT(SQLCODE);
       WHEN(Ø);
       WHEN(1ØØ) DO;
                          /* Previous record does not exist, or it has
                             already been 'cancelled',
                             result is the same - an answer string
                             'cancelled' goes to a client */

         answer_string='Canceled';
         CALL SEND_ANSWER_STRING;
       END;
       OTHER CALL ERROR_MESSAGE('Update', SQLCODE);
     END;
     /*********************************************************/
     /* Case of a VSAM file                                   */
     /*********************************************************/

transaction_record.account_number=received_client_data.account_number;

transaction_record.datetime_of_sending=received_client_data.datetime_of_sending;
     EXEC CICS READ FILE(transaction_file) INTO(transaction_record)
GENERIC
RIDFLD(generic_transaction_record_key)
KEYLENGTH(length of account_number + 17)
       UPDATE RESPONSE(response_value);
     SELECT(response_value);
       WHEN(Ø);
                    /* if there is more than one record with this generic
                       key, those are already 'cancelled' since the
                       possibility of duplicate string is avoided */
       WHEN(DFHRESP(NOTFND)) DO;  /* previous does not exist */
         answer_string='Canceled';
         CALL SEND_ANSWER_STRING;
       END;
       OTHER CALL ERROR_MESSAGE('Read update', response_value);
     END;
     IF transaction_record.do_cancel_flag='1' THEN
     DO;
       transaction_record.do_cancel_flag='Ø';
       EXEC CICS REWRITE FILE(transaction_file) FROM(transaction_record)
```

```
            RESPONSE(response_value);
      IF response_value¬=Ø THEN CALL ERROR_MESSAGE('Rewrite',
response_value);
    END;
    ELSE DO;  /* already cancelled */
      answer_string='Canceled';
      CALL SEND_ANSWER_STRING;
    END;
  END CANCEL_PREVIOUS_RECORD;

  UPDATE_FILES: PROC;
    /*************************************************************/
    /* Updates an account_balance in account_file and writes a */
    /* record into transaction_file.                             */
    /*************************************************************/
    /* Case of DB2 tables                                        */
    /*************************************************************/
    EXEC SQL SELECT account_balance
             INTO :account_record.account_balance
             FROM account_file
             WHERE account_number=:received_client_data.account_number;
    IF SQLCODE¬=Ø THEN CALL ERROR_MESSAGE('Select', SQLCODE);
    CALL DEBIT_CREDIT;

    EXEC SQL UPDATE account_file
             SET account_balance=:account_record.account_balance
             WHERE account_number=:received_client_data.account_number;
    IF SQLCODE¬=Ø THEN CALL ERROR_MESSAGE('Update', SQLCODE);

    /*****************************************************************/
    /* Calculation of a max_unique_transaction_number. If date in a */
    /* largest unique_transaction_number is different from the      */
    /* current_date, it is the first number for the current_date,   */
    /* else a transaction_counter is incremented.                   */
    /*****************************************************************/
    indmax=Ø;
    EXEC SQL SELECT MAX(unique_transaction_number)
             INTO(transaction_file.unique_transaction_number) :indmax
             FROM transaction_file;
    IF SQLCODE¬=Ø THEN CALL ERROR_MESSAGE('Select MAX', SQLCODE);

    IF indmax¬=Ø  ! (indmax=Ø &
max_unique_transaction_number.date¬=current_date)  THEN
    DO;
           /* first record at all, or first record for a current_date */
      max_unique_transaction_number.date=current_date;
      max_unique_transaction_number.transaction_counter=1;
    END;
    IF indmax=Ø & max_unique_transaction_number.date=current_date
```

```
     THEN max_unique_transaction_number.transaction_counter=
          max_unique_transaction_number.transaction_counter+1;


transaction_record.account_number=received_client_data.account_number;

transaction_record.datetime_of_sending=received_client_data.datetime_of_sending;
     transaction_record.amount=received_client_data.amount;

transaction_record.debit_credit_flag=received_client_data.debit_credit_flag;

transaction_record.current_account_balance=account_record.account_balance;

transaction_record.do_cancel_flag=received_client_string.do_cancel_flag;

     EXEC SQL INSERT INTO transaction_file VALUES
       (:transaction_record.account_number, :
transaction_record.datetime_of_sending,
        : transaction_record.unique_transaction_number, :
transaction_record.amount,
        : transaction_record.debit_credit_flag, :
transaction_record.current_account_balance,
        : transaction_record.do_cancel_flag);
     IF SQLCODE¬=Ø THEN CALL ERROR_MESSAGE('Insert', SQLCODE);
     /*********************************************************/
     /* Case of a VSAM files                                  */
     /*********************************************************/
     EXEC CICS READ FILE(account_file) INTO(account_record)
RIDFLD(account_record_key) UPDATE             RESPONSE(response_value);
     IF response_value¬=Ø THEN CALL ERROR_MESSAGE('Read update',
response_value);
     CALL DEBIT_CREDIT;
     EXEC CICS REWRITE FILE(account_file) FROM(account_record)
RESPONSE(response_value);
     IF response_value¬=Ø THEN CALL ERROR_MESSAGE('Rewrite',
response_value);
     /*********************************************************/
     /* Calculation of a max_unique_transaction_number in a case */
     /* of a VSAM file could be done in many ways: by writing a  */
     /* record with largest possible unique_transaction_number=  */
     /* 999999999999999 for each account_number and using STARTBR*/
     /* and READPREV commands, by creating an alternate index    */
     /* file with unique_transaction_number as a key etc. In this*/
     /* example the first record of a VSAM transaction_file will */
     /* have the unique transaction_record.account_number=Ø,     */
     /* transaction_record.datetime_of_sending=Ø, and            */
     /* transaction_record.unique_transaction_number field       */
     /* will keep the last and largest transaction_number.       */
     /* Of course this record should be the first record written */
```

```
    /* into a transaction_file.                                    */
     /*************************************************************/
     transaction_record.account_number=0;
     transaction_record.datetime_of_sending=0;
     EXEC CICS READ FILE(transaction_file) INTO(transaction_record)
RIDFLD(transaction_record_key) GENERIC KEYLENGTH(length of
account_number + 17)
           UPDATE RESPONSE(response_value);
     IF response_value¬=0 THEN CALL ERROR_MESSAGE('Read update first',
response_value);
     EXEC CICS DELETE FILE(transaction_file)
RIDFLD(transaction_record_key)
RESPONSE(response_value);
     IF response_value¬=0 THEN CALL ERROR_MESSAGE('Delete',
response_value);
     IF max_unique_transaction_number.date¬=current_date THEN
     DO
       max_unique_transaction_number.date=current_date;
       max_unique_transaction_number.transaction_counter=1;
     END;
     ELSE max_unique_transaction_number.transaction_counter=
          max_unique_transaction_number.transaction_counter+1;
     EXEC CICS WRITE FILE(transaction_file) FROM(transaction_record)
RIDFLD
       (transaction_record_key) RESPONSE(response_value);
     IF response_value¬=0 THEN CALL ERROR_MESSAGE('Write first',
response_value);


transaction_record.account_number=received_client_data.account_number;

transaction_record.datetime_of_sending=received_client_data.datetime_of_sending;
     transaction_record.amount=received_client_data.amount;

transaction_record.debit_credit_flag=received_client_data.debit_credit_flag;

transaction_record.current_account_balance=account_record.account_balance;

transaction_record.do_cancel_flag=received_client_string.do_cancel_flag;
     EXEC CICS WRITE FILE(transaction_file) FROM(transaction_record)
RIDFLD
       (transaction_record_key) RESPONSE(response_value);
     IF response_value¬=0 THEN CALL ERROR_MESSAGE('Write',
response_value);
  END UPDATE_FILES;

  SEND_ANSWER_STRING: PROC;

/*********************************************************************/
```

```
/* Sends an answer_string back to a client. It is useful to make a    */
/* distinction between successful and unexpectedly-ended transactions.*/
/* Every answer_string should be preceded with a character            */
/* indicating whether an answer_string is 'regular' or not. When an   */
/* answer_string is already initiated, some error or irregular        */
/* condition is raised, except if it is 'Cancelled'.                  */
/********************************************************************/

    SELECT(answer_string);
      WHEN ('Canceled')  answer_string='1'!!answer_string;
      WHEN('') answer_string='1'!!account_record.account_balance!!
                        unique_transaction_number;      /* successful */
       OTHER answer_string='Ø'!!answer_string;        /* unsuccessful */
    END;

                              /* General format of a SEND command.
                                 Depending on type of communication */
    /* (APPC, hllapi ..) it has slight differences.                  */
    EXEC CICS SEND FROM(answer_string) ERASE;
    EXEC CICS RETURN;
  END SEND_ANSWER_STRING;

  ERROR_MESSAGE: PROC(error_command, error_number);

/********************************************************************/
/* Something unpredictable has happened and SQLCODE or response_value */
/* gives the precise information. If it is not convenient to pass    */
/* such a detailed answer to an end user on a client side, then a    */
/* general answer like 'Error in program' should be sent, and an     */
/* error_record stored in order to help trace a program's            */
/* execution.                                                        */
/********************************************************************/

    DCL error_command CHAR(2Ø) VAR;
    DCL error_number  BIN FIXED(31);
    EXEC CICS SYNCPOINT ROLLBACK;
    answer_string= error_command !! ' error. Error number = ' !!
PIC(error_number);
    CALL SEND_ANSWER_STRING;
  END ERROR_MESSAGE;

  DEBIT_CREDIT: PROC;
    /*****************************************************/
    /* Depending on received_client_data.debit_credit_flag */
    /* and received_client_string.do_cancel_flag it        */
    /* calculates a new account_record.account_balance     */
    /*****************************************************/
    IF received_client_data.debit_credit_flag='+' THEN
    DO;
      IF received_client_string.do_cancel_flag='1'
```

```
      THEN
account_record.account_balance=account_record.account_balance
               +received_client_data.ammount;
      ELSE
account_record.account_balance=account_record.account_balance
               -received_client_data.ammount;
    END;
    IF received_client_data.debit_credit_flag='-' THEN
    DO;
      IF received_client_string.do_cancel_flag='1'
      THEN
account_record.account_balance=account_record.account_balance
               -received_client_data.ammount;
      ELSE
account_record.account_balance=account_record.account_balance
               +received_client_data.ammount;
    END;
  END DEBIT_CREDIT;
```

CONCLUSION

Every string in a client/server data flow should have its own unique identification.

Every record stored after a successfully-ended transaction, on either the client or CICS side, should have information from both sides as a part of its record description. That way it is possible to write a CICS server program capable of handling requests to cancel any previous action.

This condition must be met in order to write a client/server application that provides full data flow consistency.

Vladimir Antonijevic
IT Analyst
Postal Savings Bank (Yugoslavia)

To discard definitions in CICS

To discard a definition of CICS, it is necessary to use the transaction CECI, eg:

```
CECI DISCARD Resource (xxxxxxxx)
```

But this can be inconvenient because it discards only a single definition each time; and if you want to eliminate a family or group of definitions, it becomes very slow and tiresome.

The only solution is to remove from the CSD the list of definitions to be eliminated. You then have to stop CICS and perform a COLD restart, which can quite often be a big problem.

To make it easier to discard families or groups of definitions, I have written a program using the CICS System Programming Interface (SPI) commands. The program is also useful for discarding a single resource.

The resource, or resources, are discarded in two steps.

First, with the following command, the program checks that the resources exist:

```
EXEC CICS INQUIRE Resource START AT(xxxxxxxx)
```

It then uses successive INQUIRE Resource(xxxxxxxx) NEXT, until all the requested chain of characters has been checked. It then shows on the screen the first 56 resources that begin with the requested characters. At the terminals you cannot use the option START AT, because the list is not in order; therefore it is necessary to read the whole list on the terminal, up to the last entry.

Second, the resources shown are discarded, if the user confirms that is what he wants. Using F1, users can confirm which ones to discard. F3 is used to cancel any activity on a resource. If there are more than 56 resources, it is necessary to repeat the process for each group of 56. In the case of terminals, the first step OUTSERVICEs the terminals, because you cannot discard a

terminal in service, and then it discards them. It cannot put a terminal out of service and discard it in the same execution of the program because it gives the message DFHZC5916 – CICS cannot delete resource termid because DFHZCP activity is pending for this terminal.

The two steps are necessary for security because resources to be discarded are selected using the first characters of the name of the resource, and it is very easy for resources belonging to another application or software product to be included in the selection. For example, if we want to discard programs that begin with ‘AD’, we will find that GDDM programs, which begin ‘ADM’, will also be included in the list to be discarded.

For the program, I have chosen to discard terminals, transactions, program-maps, and DB2trans resource types because, in my case, they are the ones I need most often.

The program starts with the transaction XZDI and the MAPSET used is XZDI.

In the MAPSET there are four fields to fill in and several messages to be answered.

In the case of discarding a single resource you have to enter:

- Name of the resource – this is the complete name of the resource to discard.
- Type – enter terms, trans, progs, or db2tr.

When discarding a group of resources you have to enter:

- Name of the resource – the initial characters of the resources to discard.
- Type – enter terms, trans, progs, or db2tr.
- Name generic – a G indicates that a group is to be discarded.
- Length – the number of initial characters of the name of the resource.

In the first step the program returns the resources found in the

field *Resources to Try*.

The MAPSET XZDI in the first SEND MAP looks like:

```
XZDI                    DISCARDS OF DEFINITIONS IN CICS


    NAME OF THE RESOURCE:    TYPE:       TERMS (TERMINALS)
                                         TRANS (TRANSACTIONS)
         IF GENERIC NAME:   (G)          PROGS (PROGRAMS , MAPSETS)
                                         DB2TR (DB2TRANS)
                  LENGTH:   (BETWEEN 1 AND 7)



    RESOURCES TO TRY:
```

PROGRAM PXZDISD

```
      *---------------------------------------
       IDENTIFICATION DIVISION.
      * ---------------------------------------
      *   DISCARDING DEFINITIONS IN CICS
      * ---------------------------------------
      *  FIRST STEP.
      *   VALIDATE THE INPUT FIELDS AND CHECK
      *   WHETHER THE RESOURCES EXIST.
      *   THE MAP IS SENT SO THAT THE USER CONFIRMS THE ONE TO DISCARD
      *   WITH F1 OR HE DENIES IT WITH F3.
      *  SECOND STEP.
      *   TO DISCARD THE RESOURCES IF IT IS ACCEPTED
      * ---------------------------------------------------------------
      *  RESOURCES TO TRY :
      *        - TERMINALS
      *           * THE TERMINAL IS PUT OUTSERVICE IN THE FIRST STEP
      *             TO AVOID THE MESSAGE DFHZC5916
      *           * FOR TERMINALS THE INQUIRE START AT(..) IS NOT
      *             ADMITTED, THEREFORE THE INQUIRE IS NOT COMPLETE
      *             UNTIL THE END
      *        - TRANSACTIONS
      *        - PROGRAMS AND MAPSETS
      *        - DB2TRANS
      * ---------------------------------------------------------------
      *   THE PROGRAM START WITH THE TRANSACCIO XZDI
      *               IT USES THE MAPSET XZDI
      * ---------------------------------------------------------------
       PROGRAM-ID. PXZDISD.
```

```
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
COPY XZDI.
COPY RESPCICS.
COPY LDAT.
Ø1  CAMPS-TREBALL.
    Ø3  FILLER                  PIC X(7)  VALUE 'TREBALL'.
* ---------------------------------------------------------------
    Ø3  LLARG                   PIC S999  COMP VALUE +Ø.
        88  LLARG-BO      VALUE 1 THRU 7.
    Ø3  LLARG-R                 PIC 9     VALUE ZERO.
        88  LLARG-R-4     VALUE 1 THRU 4.
        88  LLARG-R-8     VALUE 1 THRU 8.
    Ø3  TIPUS                   PIC X(5)  VALUE SPACES.
        88  TIPUS-BO      VALUE 'TERMS' 'TRANS' 'DB2TR' 'PROGS'.
        88  TIPUS-4       VALUE 'TERMS' 'TRANS'.
        88  TIPUS-8       VALUE 'DB2TR' 'PROGS'.
    Ø3  RECURS-MAP         VALUE SPACES.
        Ø5  RECURS-4-MAP        PIC X(4).
        Ø5  FILLER              PIC X(4).
    Ø3  RECURS-INQ         VALUE SPACES.
        Ø5  RECURS-4-INQ        PIC X(4).
        Ø5  FILLER              PIC X(4).
    Ø3  RECURS                  PIC X(8)  VALUE SPACES.
    Ø3  RECURS-T  REDEFINES RECURS.
        Ø5  RECURS-E            PIC X  OCCURS 8 TIMES
                INDEXED BY R.
    Ø3  MISS-BO             PIC X(6Ø)  VALUE SPACES.
    Ø3  MISS-ER               VALUE SPACES.
        Ø5  MISS-ER-1           PIC X(26).
        Ø5  MISS-ER-2           PIC X(26).
        Ø5  MISS-ER-RESP2       PIC 9(8).
    Ø3  VAL-BONA                PIC X      VALUE 'S'.
    Ø3  SERV-STATUS             PIC S9(8) COMP.
        88  INSERVICE         VALUE +73.
        88  OUTSERVICE        VALUE +74.
    Ø3  CONT-R                  PIC 9(3)   VALUE ZEROS.
    Ø3  LLINIES-RECURSOS       VALUE SPACES.
        Ø5  LLINIA-1            PIC X(72).
        Ø5  LLINIA-2            PIC X(72).
        Ø5  LLINIA-3            PIC X(72).
        Ø5  LLINIA-4            PIC X(72).
        Ø5  LLINIA-5            PIC X(72).
        Ø5  LLINIA-6            PIC X(72).
        Ø5  LLINIA-7            PIC X(72).
    Ø3  TAULA-LLINIES REDEFINES LLINIES-RECURSOS.
        Ø5  T-LLINIES   OCCURS 56 TIMES   INDEXED BY LR.
            Ø7  L-RECURS        PIC X(8).
            Ø7  FILLER          PIC X.
```

```
    Ø3  NOM-CICS                PIC X(8)   VALUE SPACES.
    Ø3  SYS-CICS                PIC X(4)   VALUE SPACES.
    Ø3  FILLER                  PIC X(9)  VALUE 'FITREBALL'.
* ---------------------------------------------------------------
 PROCEDURE DIVISION.
     EXEC CICS ASSIGN APPLID(NOM-CICS)
                      SYSID(SYS-CICS)
                      END-EXEC.
     EXEC CICS RECEIVE MAP('XZDIX') MAPSET('XZDI')
          NOHANDLE END-EXEC.
     IF  EIBRESP = RESP-MAPFAIL OR EIBAID = 3
         EXEC CICS SEND MAP('XZDIX') MAPSET('XZDI') MAPONLY
              ERASE END-EXEC
     ELSE
         MOVE SPACES TO CONTO  MISSBO  MISSEO
         IF  EIBAID = '1' AND CONFI = 'S'
             MOVE LR1O  TO  LLINIA-1
             MOVE LR2O  TO  LLINIA-2
             MOVE LR3O  TO  LLINIA-3
             MOVE LR4O  TO  LLINIA-4
             MOVE LR5O  TO  LLINIA-5
             MOVE LR6O  TO  LLINIA-6
             MOVE LR7O  TO  LLINIA-7
         ELSE
             MOVE SPACES TO LR1O LR2O LR3O LR4O LR5O LR6O LR7O
         END-IF
         PERFORM VALIDAR-ENTRADA
         IF  VAL-BONA = 'S'
             PERFORM PROCES
             IF  CONT-R > Ø
                 MOVE CONT-R  TO CONTO
             END-IF
             MOVE MISS-BO TO MISSBO
             MOVE MISS-ER TO MISSEO
             EXEC CICS SEND MAP('XZDIX') MAPSET('XZDI')
                  DATAONLY CURSOR END-EXEC
         ELSE
             EXEC CICS SEND MAP('XZDIX') MAPSET('XZDI')
                  DATAONLY CURSOR END-EXEC
         END-IF
     END-IF.
     EXEC CICS RETURN END-EXEC.
     GOBACK.
* ---------------------------------------------------------------
 PROCES.
     MOVE RECURSI TO RECURS-MAP
     IF  TIPUS = 'TERMS'
       IF  GENERI = 'G'
           IF  EIBAID = '1' AND CONFI = 'S'
               PERFORM PROCES-G-RECURS-DIS
```

```
            ELSE
                PERFORM PROCES-G-TERMS-INQ-OUT
            END-IF
          ELSE
            IF  EIBAID = '1' AND CONFI = 'S'
                PERFORM PROCES-TERM-DIS
            ELSE
                PERFORM PROCES-TERM-OUT
            END-IF
          END-IF
        ELSE
          IF  GENERI = 'G'
            IF  EIBAID = '1' AND CONFI = 'S'
                PERFORM PROCES-G-RECURS-DIS
            ELSE
                PERFORM PROCES-G-RECURS-INQ
            END-IF
          ELSE
            IF  EIBAID = '1' AND CONFI = 'S'
                PERFORM PROCES-RECURS-DIS
            ELSE
                PERFORM PROCES-RECURS-INQ
            END-IF
          END-IF
        END-IF.
      * ---------------------------------------------------------------
      * TO TRY TERMINALS
      * ---------------------------------------------------------------
       PROCES-TERM-OUT.
           MOVE RECURS-4-MAP TO RECURS-4-INQ.
           PERFORM INQUIRE-RECURS.
           IF  EIBRESP = RESP-OK
               PERFORM OUTSERVICE-TERMINAL
           END-IF.
           IF  EIBRESP = RESP-OK
               MOVE 'TERMINAL PUT OUTSERVICE' TO MISS-BO
               MOVE 'TO CONFIRM WITH F1 THE DISCARD OR F3 TO REJECT'
                    TO MISS-ER
              MOVE 'S' TO CONFO
               MOVE  AT-PROT-MOD TO TIPUSA RECURSA GENERA LLARGA
           ELSE
               MOVE 'RESOURCE NOT FOUND' TO MISS-ER
           END-IF.
       PROCES-TERM-DIS.
           MOVE RECURS-4-MAP TO RECURS-4-INQ.
           PERFORM DISCARD-RECURS
           IF  EIBRESP = RESP-OK
               MOVE 'DISCARDED RESOURCE' TO MISS-ER
              MOVE 'N' TO CONFO
               MOVE  AT-ALFA-MOD TO TIPUSA RECURSA GENERA LLARGA
```

```
           ELSE
               MOVE  RECURS-4-MAP              TO MISS-ER-1
               MOVE 'ERROR EN DISCARD RESP2 :' TO MISS-ER-2
               MOVE  EIBRESP2                  TO MISS-ER-RESP2
           END-IF.
       PROCES-G-TERMS-INQ-OUT.
           MOVE RECURS-MAP TO RECURS-INQ.
           PERFORM INQUIRE-START.
           PERFORM INQUIRE-RECURS-NEXT.
           SET LR TO 1.
           PERFORM TRACTAR-G-TERMS-INQ-OUT UNTIL EIBRESP   = RESP-END
                                           OR          LR > 56
                                           OR EIBRESP NOT = RESP-OK.
           PERFORM INQUIRE-END.
           IF  CONT-R = Ø
               MOVE 'RESOURCES NOT FOUND ' TO MISS-ER
           ELSE
               IF  LR > 56
                   MOVE
                    'TERMINALS PUT OUTSERVICE, MORE THAN 56 TO TRY'
                     TO MISS-BO
               ELSE
                   MOVE
                    'TERMINALS PUT OUTSERVICE'
                     TO MISS-BO
               END-IF
               MOVE 'TO CONFIRM WITH F1 THE DISCARD OR F3 TO REJECT'
                   TO MISS-ER
               MOVE 'RESOURCES TO TRY :'      TO MTRATO
               MOVE 'S' TO CONFO
               MOVE AT-PROT-MOD TO TIPUSA RECURSA GENERA LLARGA
               MOVE LLINIA-1  TO  LR1O
               MOVE LLINIA-2  TO  LR2O
               MOVE LLINIA-3  TO  LR3O
               MOVE LLINIA-4  TO  LR4O
               MOVE LLINIA-5  TO  LR5O
               MOVE LLINIA-6  TO  LR6O
               MOVE LLINIA-7  TO  LR7O
           END-IF.
       TRACTAR-G-TERMS-INQ-OUT.
           IF  RECURS(1:LLARG) = RECURS-INQ(1:LLARG)
               PERFORM OUTSERVICE-TERMINAL
               IF EIBRESP = RESP-OK
                   MOVE RECURS-INQ  TO L-RECURS(LR)
                   ADD 1 TO CONT-R
                   SET LR UP BY 1
               END-IF
           END-IF.
           PERFORM INQUIRE-RECURS-NEXT.
      * ---------------------------------------------------------------
```

```
* TO TRY TRANSACTIONS, PROGRAMS, MAPSET, DB2TRANS
* -------------------------------------------------------------
 PROCES-RECURS-INQ.
     MOVE RECURS-MAP TO RECURS-INQ.
     PERFORM INQUIRE-RECURS.
     IF  EIBRESP = RESP-OK
         MOVE 'TO CONFIRM WITH F1 THE DISCARD OR F3 TO REJECT'
              TO MISS-ER
         MOVE 'S' TO CONFO
         MOVE  AT-PROT-MOD TO TIPUSA RECURSA GENERA LLARGA
     ELSE
         MOVE 'RESOURCE NOT FOUND' TO MISS-ER
     END-IF.
 PROCES-RECURS-DIS.
     MOVE RECURS-MAP TO RECURS-INQ.
     PERFORM DISCARD-RECURS
     IF  EIBRESP = RESP-OK
         MOVE 'DISCARDED RESOURCE' TO MISS-ER
         MOVE 'N' TO CONFO
         MOVE  AT-ALFA-MOD TO TIPUSA RECURSA GENERA LLARGA
     ELSE
        IF TIPUS = 'PROGS' AND EIBRESP2 = 11
          MOVE  RECURS-MAP                TO MISS-ER-1
          MOVE 'PROGRAM O MAPSET IN USE  ' TO MISS-ER-2
          MOVE  EIBRESP2                  TO MISS-ER-RESP2
        ELSE
          MOVE  RECURS-MAP                TO MISS-ER-1
          MOVE 'ERROR EN DISCARD RESP2 :' TO MISS-ER-2
          MOVE  EIBRESP2                  TO MISS-ER-RESP2
        END-IF
     END-IF.
* -------------------------------------------------------------
 PROCES-G-RECURS-INQ.
     MOVE RECURS-MAP TO RECURS-INQ.
     PERFORM INQUIRE-START.
     PERFORM INQUIRE-RECURS-NEXT.
     SET LR TO 1.
     PERFORM TRACTAR-G-RECURS-INQ UNTIL  EIBRESP = RESP-END
                    OR  RECURS-INQ(1:LLARG) > RECURS-MAP(1:LLARG)
                                  OR          LR > 56
                                  OR EIBRESP NOT = RESP-OK.
     PERFORM INQUIRE-END.
     IF  CONT-R = Ø
         MOVE 'RESOURCES NOT FOUND ' TO MISS-ER
     ELSE
         IF  LR > 56
             MOVE 'MORE THAN 56 RESOURCES TO TRY' TO MISS-BO
         END-IF
         MOVE 'TO CONFIRM WITH F1 THE DISCARD OR F3 TO REJECT'
             TO MISS-ER
```

```
            MOVE 'RESOURCES TO TRY :'      TO MTRATO
            MOVE 'S' TO CONFO
            MOVE AT-PROT-MOD TO TIPUSA RECURSA GENERA LLARGA
            MOVE LLINIA-1  TO  LR1O
            MOVE LLINIA-2  TO  LR2O
            MOVE LLINIA-3  TO  LR3O
            MOVE LLINIA-4  TO  LR4O
            MOVE LLINIA-5  TO  LR5O
            MOVE LLINIA-6  TO  LR6O
            MOVE LLINIA-7  TO  LR7O
        END-IF.
     TRACTAR-G-RECURS-INQ.
        IF  RECURS-MAP(1:LLARG) = RECURS-INQ(1:LLARG)
            MOVE RECURS-INQ TO L-RECURS(LR)
            ADD 1 TO CONT-R
            SET LR UP BY 1
        END-IF.
        PERFORM INQUIRE-RECURS-NEXT.
    * ---------------------------------------------------------------
     PROCES-G-RECURS-DIS.
        SET LR TO 1.
        PERFORM TRACTAR-G-RECURS-DIS UNTIL          LR > 56
                                       OR L-RECURS(LR) = SPACES
                                       OR  EIBRESP NOT = RESP-OK.
        IF  CONT-R = Ø
            MOVE 'RESOURCES NOT FOUND ' TO MISS-ER
        ELSE
            MOVE 'DISCARDED RESOURCES'     TO MISS-ER
            MOVE 'TREATED RESOURCES :'      TO MTRATO
        END-IF.
        MOVE 'N' TO CONFO.
        MOVE AT-ALFA-MOD TO TIPUSA RECURSA GENERA LLARGA.
     TRACTAR-G-RECURS-DIS.
        MOVE L-RECURS(LR) TO RECURS-INQ.
        PERFORM DISCARD-RECURS.
        IF  EIBRESP = RESP-OK
            ADD 1 TO CONT-R
            SET LR UP BY 1
        END-IF.
    * ---------------------------------------------------------------
    * TO VALIDATE INPUT DATA
    * ---------------------------------------------------------------
     VALIDAR-ENTRADA.
        MOVE RECURSI TO RECURS.
        MOVE TIPUSI TO TIPUS.
        IF  RECURSI = SPACES
          MOVE 'N'  TO  VAL-BONA
          MOVE 'IT LACKS THE NAME OF RESOURCE' TO MISSEO
          MOVE -1  TO  RECURSL
        ELSE
```

```
 PERFORM CONTAR-LLARG-RECURS VARYING R FROM 1 BY 1
          UNTIL RECURS-E(R) = SPACE OR R > 8
 IF  RECURSI(1:1) = 'C' OR RECURSI(1:3) = 'DFH'
    MOVE 'N'  TO  VAL-BONA
    MOVE 'RESOURCE NAME PROTECTED, IS A RESOURCE OF CICS'
           TO MISSEO
    MOVE -1  TO  RECURSL
 ELSE
  IF  NOT TIPUS-BO
    MOVE 'N'  TO  VAL-BONA
    MOVE 'TYPE OF RESOURCE, NOT VALID' TO MISSEO
    MOVE -1  TO  TIPUSL
  ELSE
   IF  GENERI NOT = SPACES
    IF  GENERI NOT = 'G'
      MOVE 'N'  TO  VAL-BONA
      MOVE 'IF GENERIC NAME IT SHOULD BE = G' TO MISSEO
      MOVE -1  TO  GENERL
    ELSE
     IF  LLARGI NOT NUMERIC
       MOVE 'N'  TO  VAL-BONA
       MOVE 'LENGTH NOT NUMERIC' TO MISSEO
       MOVE -1  TO  LLARGL
     ELSE
      MOVE LLARGI  TO  LLARG
      IF  NOT LLARG-BO
        MOVE 'N'  TO  VAL-BONA
        MOVE 'LENGTH NOT VALID ' TO MISSEO
        MOVE -1  TO  LLARGL
      END-IF
     END-IF
    END-IF
   END-IF
  END-IF
 END-IF
END-IF.
IF  VAL-BONA = 'S'
 IF  TIPUS-4 AND LLARG-R > 4
    MOVE 'N'  TO  VAL-BONA
    MOVE 'NAME OF RESOURCE NOT VALID, > 4 CHARACTERS'
        TO MISSEO
    MOVE -1  TO  RECURSL
 ELSE
  IF  GENERI = 'G'
   IF  TIPUS-4 AND LLARG-R > 3
     MOVE 'N'  TO  VAL-BONA
     MOVE
         'IF GENERIC, NAME IT SHOULD BE MAX. 3 CHARACTERS'
          TO MISSBO
     MOVE 'FOR TERMINALS AND TRANSACTIONS'
```

```
                TO MISSEO
            MOVE -1  TO  RECURSL
          ELSE
           IF  TIPUS-8 AND LLARG-R > 7
             MOVE 'N'  TO  VAL-BONA
             MOVE
              'IF GENERIC, NAME IT SHOULD BE MAX. 7 CARACTERES'
                 TO MISSBO
              MOVE 'FOR PROGRAMS, MAPSET AND DB2TRANS'
                 TO MISSEO
             MOVE -1  TO  RECURSL
           ELSE
            IF  LLARG-R NOT = LLARG
              MOVE 'N'  TO  VAL-BONA
              MOVE 'LENGTH NOT EQUAL TO LENGTH OF NAME'
                  TO MISSEO
              MOVE -1  TO  LLARGL
            END-IF
           END-IF
          END-IF
         END-IF
        END-IF
       END-IF.
 CONTAR-LLARG-RECURS.
       IF  RECURS-E(R) NOT = SPACE
           ADD 1 TO LLARG-R
       END-IF.
* ---------------------------------------------------------------
* SYSTEM COMMANDS
* ---------------------------------------------------------------
 OUTSERVICE-TERMINAL.
       EXEC CICS SET TERMINAL(RECURS-4-INQ)
                 OUTSERVICE
                 NOHANDLE END-EXEC.
 INQUIRE-RECURS.
       IF  TIPUS = 'TERMS'
           EXEC CICS INQUIRE TERMINAL(RECURS-4-INQ)
                SERVSTATUS(SERV-STATUS)
                NOHANDLE END-EXEC
       END-IF.
       IF  TIPUS = 'TRANS'
           EXEC CICS INQUIRE TRANSACTION(RECURS-4-INQ)
                NOHANDLE END-EXEC
       END-IF.
       IF  TIPUS = 'PROGS'
           EXEC CICS INQUIRE PROGRAM(RECURS-INQ)
                NOHANDLE END-EXEC
       END-IF.
       IF  TIPUS = 'DB2TR'
           EXEC CICS INQUIRE DB2TRAN(RECURS-INQ)
```

```
                NOHANDLE END-EXEC
        END-IF.
    INQUIRE-RECURS-NEXT.
        IF  TIPUS = 'TERMS'
            EXEC CICS INQUIRE TERMINAL(RECURS-4-INQ) NEXT
                 SERVSTATUS(SERV-STATUS)
                 NOHANDLE END-EXEC
        END-IF.
        IF  TIPUS = 'TRANS'
            EXEC CICS INQUIRE TRANSACTION(RECURS-4-INQ) NEXT
                 NOHANDLE END-EXEC
        END-IF.
        IF  TIPUS = 'PROGS'
            EXEC CICS INQUIRE PROGRAM(RECURS-INQ) NEXT
                 NOHANDLE END-EXEC
        END-IF.
        IF  TIPUS = 'DB2TR'
            EXEC CICS INQUIRE DB2TRAN(RECURS-INQ) NEXT
                 NOHANDLE END-EXEC
        END-IF.
    DISCARD-RECURS.
        IF  TIPUS = 'TERMS'
            EXEC CICS DISCARD TERMINAL(RECURS-4-INQ)
                 NOHANDLE END-EXEC
        END-IF.
        IF  TIPUS = 'TRANS'
            EXEC CICS DISCARD TRANSACTION(RECURS-4-INQ)
                 NOHANDLE END-EXEC
        END-IF.
        IF  TIPUS = 'PROGS'
            EXEC CICS DISCARD PROGRAM(RECURS-INQ)
                 NOHANDLE END-EXEC
        END-IF.
        IF  TIPUS = 'DB2TR'
            EXEC CICS DISCARD DB2TRAN(RECURS-INQ)
                 NOHANDLE END-EXEC
        END-IF.
    INQUIRE-START.
        IF  TIPUS = 'TERMS'
            EXEC CICS INQUIRE TERMINAL    START
                  NOHANDLE END-EXEC
        END-IF.
        IF  TIPUS = 'TRANS'
            EXEC CICS INQUIRE TRANSACTION START AT(RECURS-4-INQ)
                  NOHANDLE END-EXEC
        END-IF.
        IF  TIPUS = 'PROGS'
            EXEC CICS INQUIRE PROGRAM     START AT(RECURS-INQ)
                  NOHANDLE END-EXEC
        END-IF.
```

```
           IF  TIPUS = 'DB2TR'
               EXEC CICS INQUIRE DB2TRAN     START AT(RECURS-INQ)
                     NOHANDLE END-EXEC
           END-IF.
       INQUIRE-END.
           IF  TIPUS = 'TERMS'
               EXEC CICS INQUIRE TERMINAL    END
                     NOHANDLE END-EXEC
           END-IF.
           IF  TIPUS = 'TRANS'
               EXEC CICS INQUIRE TRANSACTION END
                     NOHANDLE END-EXEC
           END-IF.
           IF  TIPUS = 'PROGS'
               EXEC CICS INQUIRE PROGRAM     END
                     NOHANDLE END-EXEC
           END-IF.
           IF  TIPUS = 'DB2TR'
               EXEC CICS INQUIRE DB2TRAN     END
                     NOHANDLE END-EXEC
           END-IF.
```

XZDI SOURCE BMS

```
XZDI     DFHMSD MODE=INOUT,LANG=COBOL,TIOAPFX=YES,TYPE=&SYSPARM,       X
               CTRL=FREEKB
XZDIX    DFHMDI SIZE=(22,80)
PANTTR   DFHMDF POS=(01,01),LENGTH=04,ATTRB=(ASKIP,FSET),             X
               INITIAL='XZDI'
         DFHMDF POS=(01,20),LENGTH=40,ATTRB=(ASKIP,BRT),              X
               INITIAL='DISCARDS OF DEFINITIONS IN CICS'
CONF     DFHMDF POS=(01,63),LENGTH=01,ATTRB=(PROT,FSET,DRK)
*
         DFHMDF POS=(05,02),LENGTH=22,ATTRB=ASKIP,                    X
               INITIAL='NAME OF THE RESOURCE :'
RECURS   DFHMDF POS=(05,25),LENGTH=08,ATTRB=(UNPROT,FSET,IC),         X
               INITIAL='        '
         DFHMDF POS=(05,34),LENGTH=06,ATTRB=ASKIP,                    X
               INITIAL='TYPE :'
TIPUS    DFHMDF POS=(05,41),LENGTH=05,ATTRB=(UNPROT,FSET),            X
               INITIAL='     '
         DFHMDF POS=(05,47),LENGTH=30,ATTRB=ASKIP,                    X
               INITIAL='TERMS (TERMINALS)'
         DFHMDF POS=(06,47),LENGTH=30,ATTRB=ASKIP,                    X
               INITIAL='TRANS (TRANSACTIONS)'
*
         DFHMDF POS=(07,07),LENGTH=17,ATTRB=ASKIP,                    X
               INITIAL='IF GENERIC NAME :'
GENER    DFHMDF POS=(07,25),LENGTH=01,ATTRB=(UNPROT,FSET),            X
```

```
              INITIAL=' '
          DFHMDF POS=(Ø7,27),LENGTH=Ø3,ATTRB=ASKIP,                      X
              INITIAL='(G)'
          DFHMDF POS=(Ø7,47),LENGTH=3Ø,ATTRB=ASKIP,                      X
              INITIAL='PROGS (PROGRAMS , MAPSETS)'
*
          DFHMDF POS=(Ø8,47),LENGTH=3Ø,ATTRB=ASKIP,                      X
              INITIAL='DB2TR (DB2TRANS)'
*
          DFHMDF POS=(Ø9,16),LENGTH=Ø8,ATTRB=ASKIP,                      X
              INITIAL='LENGTH :'
LLARG     DFHMDF POS=(Ø9,25),LENGTH=Ø1,ATTRB=(UNPROT,FSET,NUM)
          DFHMDF POS=(Ø9,27),LENGTH=2Ø,ATTRB=ASKIP,                      X
              INITIAL='(BETWEEN 1 AND 7)'
*
MISSB     DFHMDF POS=(11,1Ø),LENGTH=6Ø,ATTRB=(PROT,FSET,BRT)
MISSE     DFHMDF POS=(12,1Ø),LENGTH=6Ø,ATTRB=(PROT,FSET,BRT)
*
MTRAT     DFHMDF POS=(14,Ø6),LENGTH=18,ATTRB=ASKIP,                      X
              INITIAL='RESOURCES TO TRY :'
CONT      DFHMDF POS=(14,25),LENGTH=Ø3,ATTRB=(PROT,FSET,BRT)
LR1       DFHMDF POS=(15,Ø7),LENGTH=72,ATTRB=(PROT,FSET,BRT)
LR2       DFHMDF POS=(16,Ø7),LENGTH=72,ATTRB=(PROT,FSET,BRT)
LR3       DFHMDF POS=(17,Ø7),LENGTH=72,ATTRB=(PROT,FSET,BRT)
LR4       DFHMDF POS=(18,Ø7),LENGTH=72,ATTRB=(PROT,FSET,BRT)
LR5       DFHMDF POS=(19,Ø7),LENGTH=72,ATTRB=(PROT,FSET,BRT)
LR6       DFHMDF POS=(2Ø,Ø7),LENGTH=72,ATTRB=(PROT,FSET,BRT)
LR7       DFHMDF POS=(21,Ø7),LENGTH=72,ATTRB=(PROT,FSET,BRT)
          DFHMSD TYPE=FINAL
          END
```

RESPCICS

```
      *      POSSIBLE VALUES FOR EIBRESP FIELD
      *      CICS T/S V1.3.Ø           ABRIL-2ØØØ
       Ø1  RESPCICS.
           Ø3  RESP-OK                 PIC S9(8) COMP VALUE +Ø.
      *
           Ø3  RESP-ERROR              PIC S9(8) COMP VALUE +1.
           Ø3  RESP-RDATT              PIC S9(8) COMP VALUE +2.
           Ø3  RESP-WRBRK              PIC S9(8) COMP VALUE +3.
           Ø3  RESP-EOF                PIC S9(8) COMP VALUE +4.
           Ø3  RESP-EODS               PIC S9(8) COMP VALUE +5.
           Ø3  RESP-EOC                PIC S9(8) COMP VALUE +6.
           Ø3  RESP-INBFMH             PIC S9(8) COMP VALUE +7.
           Ø3  RESP-ENDINPT            PIC S9(8) COMP VALUE +8.
           Ø3  RESP-NOVAL              PIC S9(8) COMP VALUE +9.
      *
           Ø3  RESP-NOSTART            PIC S9(8) COMP VALUE +1Ø.
```

```
      Ø3  RESP-TERMIDERR          PIC S9(8) COMP VALUE +11.
      Ø3  RESP-FILENOTFOUND       PIC S9(8) COMP VALUE +12.
      Ø3  RESP-NOTFND             PIC S9(8) COMP VALUE +13.
      Ø3  RESP-DUPREC             PIC S9(8) COMP VALUE +14.
      Ø3  RESP-DUPKEY             PIC S9(8) COMP VALUE +15.
      Ø3  RESP-INVREQ             PIC S9(8) COMP VALUE +16.
      Ø3  RESP-IOERR              PIC S9(8) COMP VALUE +17.
      Ø3  RESP-NOSPACE            PIC S9(8) COMP VALUE +18.
      Ø3  RESP-NOTOPEN            PIC S9(8) COMP VALUE +19.
*
      Ø3  RESP-ENDFILE            PIC S9(8) COMP VALUE +2Ø.
      Ø3  RESP-ILLOGIC            PIC S9(8) COMP VALUE +21.
      Ø3  RESP-LENGERR            PIC S9(8) COMP VALUE +22.
      Ø3  RESP-QZERO              PIC S9(8) COMP VALUE +23.
      Ø3  RESP-SIGNAL             PIC S9(8) COMP VALUE +24.
      Ø3  RESP-QBUSY              PIC S9(8) COMP VALUE +25.
      Ø3  RESP-ITEMERR            PIC S9(8) COMP VALUE +26.
      Ø3  RESP-PGMIDERR           PIC S9(8) COMP VALUE +27.
      Ø3  RESP-TRANSIDERR         PIC S9(8) COMP VALUE +28.
      Ø3  RESP-ENDDATA            PIC S9(8) COMP VALUE +29.
*
      Ø3  RESP-INVTSREQ           PIC S9(8) COMP VALUE +3Ø.
      Ø3  RESP-EXPIRED            PIC S9(8) COMP VALUE +31.
      Ø3  RESP-RETPAGE            PIC S9(8) COMP VALUE +32.
      Ø3  RESP-RTEFAIL            PIC S9(8) COMP VALUE +33.
      Ø3  RESP-RTESOME            PIC S9(8) COMP VALUE +34.
      Ø3  RESP-TSIOERR            PIC S9(8) COMP VALUE +35.
      Ø3  RESP-MAPFAIL            PIC S9(8) COMP VALUE +36.
      Ø3  RESP-INVERRTERM         PIC S9(8) COMP VALUE +37.
      Ø3  RESP-INVMPSZ            PIC S9(8) COMP VALUE +38.
      Ø3  RESP-IGREQID            PIC S9(8) COMP VALUE +39.
*
      Ø3  RESP-OVERFLOW           PIC S9(8) COMP VALUE +4Ø.
      Ø3  RESP-INVLDC             PIC S9(8) COMP VALUE +41.
      Ø3  RESP-NOSTG              PIC S9(8) COMP VALUE +42.
      Ø3  RESP-JIDERR             PIC S9(8) COMP VALUE +43.
      Ø3  RESP-QIDERR             PIC S9(8) COMP VALUE +44.
      Ø3  RESP-NOJBUFSP           PIC S9(8) COMP VALUE +45.
      Ø3  RESP-DSSTAT             PIC S9(8) COMP VALUE +46.
      Ø3  RESP-SELNERR            PIC S9(8) COMP VALUE +47.
      Ø3  RESP-FUNCERR            PIC S9(8) COMP VALUE +48.
      Ø3  RESP-UNEXPIN            PIC S9(8) COMP VALUE +49.
*
      Ø3  RESP-NOPASSBKRD         PIC S9(8) COMP VALUE +5Ø.
      Ø3  RESP-NOPASSBKWR         PIC S9(8) COMP VALUE +51.
*                                                  +52
      Ø3  RESP-SYSIDERR           PIC S9(8) COMP VALUE +53.
      Ø3  RESP-ISCINVREQ          PIC S9(8) COMP VALUE +54.
      Ø3  RESP-ENQBUSY            PIC S9(8) COMP VALUE +55.
      Ø3  RESP-ENVDEFERR          PIC S9(8) COMP VALUE +56.
```

```
           03  RESP-IGREQCD            PIC S9(8) COMP VALUE +57.
           03  RESP-SESSIONERR         PIC S9(8) COMP VALUE +58.
           03  RESP-SYSBUSY            PIC S9(8) COMP VALUE +59.
      *
           03  RESP-SESSBUSY           PIC S9(8) COMP VALUE +60.
           03  RESP-NOTALLOC           PIC S9(8) COMP VALUE +61.
           03  RESP-CBIDERR            PIC S9(8) COMP VALUE +62.
           03  RESP-INVEXITREQ         PIC S9(8) COMP VALUE +63.
           03  RESP-INVPARTNSET        PIC S9(8) COMP VALUE +64.
           03  RESP-INVPARTN           PIC S9(8) COMP VALUE +65.
           03  RESP-PARTNFAIL          PIC S9(8) COMP VALUE +66.
      *                                                 +67
      *                                                 +68
           03  RESP-USERIDERR          PIC S9(8) COMP VALUE +69.
      *
           03  RESP-NOTAUTH            PIC S9(8) COMP VALUE +70.
           03  RESP-VOLIDERR           PIC S9(8) COMP VALUE +71.
           03  RESP-SUPPRESSED         PIC S9(8) COMP VALUE +72.
      *                                                 +73
      *                                                 +74
           03  RESP-RESIDERR           PIC S9(8) COMP VALUE +75.
      *                                                 +76
      *                                                 +77
      *                                                 +78
      *                                                 +79
      *
           03  RESP-NOSPOOL            PIC S9(8) COMP VALUE +80.
           03  RESP-TERMERR            PIC S9(8) COMP VALUE +81.
           03  RESP-ROLLEDBACK         PIC S9(8) COMP VALUE +82.
           03  RESP-END                PIC S9(8) COMP VALUE +83.
           03  RESP-DISABLED           PIC S9(8) COMP VALUE +84.
           03  RESP-ALLOCERR           PIC S9(8) COMP VALUE +85.
           03  RESP-STRELERR           PIC S9(8) COMP VALUE +86.
           03  RESP-OPENERR            PIC S9(8) COMP VALUE +87.
           03  RESP-SPOLBUSY           PIC S9(8) COMP VALUE +88.
           03  RESP-SPOLERR            PIC S9(8) COMP VALUE +89.
      *
           03  RESP-NODEIDER           PIC S9(8) COMP VALUE +90.
           03  RESP-TASKIDERR          PIC S9(8) COMP VALUE +91.
           03  RESP-TCIDERR            PIC S9(8) COMP VALUE +92.
           03  RESP-DSNNOTFOUND        PIC S9(8) COMP VALUE +93.
           03  RESP-LOADING            PIC S9(8) COMP VALUE +94.
           03  RESP-MODELIDERR         PIC S9(8) COMP VALUE +95.
           03  RESP-OUTDESCERR         PIC S9(8) COMP VALUE +96.
           03  RESP-PARTNERIDERR       PIC S9(8) COMP VALUE +97.
           03  RESP-PROFILEIDERR       PIC S9(8) COMP VALUE +98.
           03  RESP-NETNAMEIDERR       PIC S9(8) COMP VALUE +99.
      *
           03  RESP-LOCKED             PIC S9(8) COMP VALUE +100.
           03  RESP-RECORDBUSY         PIC S9(8) COMP VALUE +101.
```

```
           03  RESP-UOWNOTFOUND        PIC S9(8) COMP VALUE +102.
           03  RESP-UOWLNOTFOUND       PIC S9(8) COMP VALUE +103.
           03  RESP-LINKABEND          PIC S9(8) COMP VALUE +104.
           03  RESP-CHANGED            PIC S9(8) COMP VALUE +105.
           03  RESP-PROCESSBUSY        PIC S9(8) COMP VALUE +106.
           03  RESP-ACTIVITYBUSY       PIC S9(8) COMP VALUE +107.
           03  RESP-PROCESSERR         PIC S9(8) COMP VALUE +108.
           03  RESP-ACTIVITYERR        PIC S9(8) COMP VALUE +109.
      *
           03  RESP-CONTAINERERR       PIC S9(8) COMP VALUE +110.
           03  RESP-EVENTERR           PIC S9(8) COMP VALUE +111.
           03  RESP-TOKENERR           PIC S9(8) COMP VALUE +112.
           03  RESP-NOTFINISHED        PIC S9(8) COMP VALUE +113.
           03  RESP-POOLERR            PIC S9(8) COMP VALUE +114.
           03  RESP-TIMERERR           PIC S9(8) COMP VALUE +115.
           03  RESP-SYMBOLERR          PIC S9(8) COMP VALUE +116.
           03  RESP-TEMPLATERR         PIC S9(8) COMP VALUE +117.
      *
```

LDAT

```
       01  AT-ATRIBUTOS.
      *................................................
      *            NUMBERS OF THE FIELDS               .
      *................................................
      * POSITION . ATTRIBUTE . DESCRIPTION             .
      *................................................
      *          .  ALFA    . ALPHABETIC NOT PROTECTED .
      *  FIRST   .  NUM     . NUMERIC NOT PROTECTED    .
      *          .  PROT    . PROTECTED                .
      *................................................
      *          .          . NORMAL                   .
      *  SECOND  .  BRI     . BRIGHT                   .
      *          .  NODISP  . NO DISPLAY (OSCURO)      .
      *................................................
      *  THIRD   .          . NO MODIFICATION          .
      *          .  MOD     . MODIFICATION             .
      *................................................
           03 AT-NO-MODIFICADO.
              05 AT-NORMDISP.
                07 AT-ALFA                PIC X VALUE ' '.
                07 AT-NUM                 PIC X VALUE '&'.
                07 AT-PROT                PIC X VALUE '0'.
              05 AT-BRI.
                07 AT-ALFA-BRI            PIC X VALUE 'H'.
                07 AT-NUM-BRI             PIC X VALUE 'Q'.
                07 AT-PROT-BRI            PIC X VALUE '8'.
              05 AT-NODISP.
                07 AT-ALFA-NODISP         PIC X VALUE '<'.
```

```
           07 AT-NUM-NODISP           PIC X VALUE '*'.
           07 AT-PROT-NODISP          PIC X VALUE '@'.
       03 AT-MODIFICADO.
          05 AT-NORMDISP-MOD.
           07 AT-ALFA-MOD             PIC X VALUE 'A'.
           07 AT-NUM-MOD              PIC X VALUE 'J'.
           07 AT-PROT-MOD             PIC X VALUE '1'.
          05 AT-BRI.
           07 AT-ALFA-BRI-MOD         PIC X VALUE 'I'.
           07 AT-NUM-BRI-MOD          PIC X VALUE 'R'.
           07 AT-PROT-BRI-MOD         PIC X VALUE '9'.
          05 AT-NODISP.
           07 AT-ALFA-NODISP-MOD      PIC X VALUE '('.
           07 AT-NUM-NODISP-MOD       PIC X VALUE ')'.
           07 AT-PROT-NODISP-MOD      PIC X VALUE QUOTE.
```

Juan Tormo
System Manager
Sidmed SA (Spain)

Why not share your expertise and earn money at the same time? *CICS Update* is looking for REXX EXECs, program code, JavaScript, etc, that experienced CICS users have written to make their life, or the lives of their users, easier. We are also looking for hints and tips that experienced users want to pass on to new CICS programmers.

We will publish your article (after vetting by our expert panel) and send you a cheque when it is published. Articles can be of any length and can be e-mailed to Trevor Eddolls at trevore@xephon.com.

If you're not sure whether an idea you have is worth turning into an article, then e-mail your idea to trevore@xephon.com and Trevor will let you know straight away.

A free copy of our *Notes for contributors*, which includes information on our payment rates, is available from our Web site at www.xephon.com/nfc.

Audit trail for CICS maxtask events – part 2

This month we conclude the code for programs that can help to show what is happening both within and across systems when maxtask events occur.

```
*-------------------------------------------------------------------*
* EXTRACT EXIT GLOBAL WORK AREA ADDRESS                             *
*-------------------------------------------------------------------*
          EXEC  CICS EXTRACT EXIT                                    X
                     PROGRAM('CZSMXTRP')                             X
                     GASET(R7)                                       X
                     GALENGTH(LENGTH)                                X
                     RESP(RESP1)                                     X
                     RESP2(RESP2)
          CLC   RESP1,DFHRESP(NORMAL)        OK?
          BNE   ERRORGA1                     NO - DEAL WITH ANY ERROR
          USING GADSECT,R7
          CLC   GALIT,=C'CZSMXTRP'           INITIALIZED
          BNE   RETURN                       NO - JUST IGNORE
*-------------------------------------------------------------------*
* SCAN THROUGH ELEMENT ARRAY TO CHECK DATES                         *
*-------------------------------------------------------------------*
          LA    R8,GALMENT
          USING GAENTRY,R8
LOOP1    EQU   *
          CLC   GADATE(8),GALPRT           CREATED SINCE LAST PRINT
          BNH   NEXT1                      NO - ALREADY DONE
          CLC   GADATE(8),CDATE            CREATED SINCE WE STARTED
          BH    NEXT1                      YES - DO IT NEXT TIME
          CLC   GADATE,=F'Ø'               SLOT NOT BEEN USED
          BE    NEXT1                      YES - CHECK NEXT
*-------------------------------------------------------------------*
* PRINT THIS ENTRY                                                  *
*-------------------------------------------------------------------*
          MVC   TDREC,TDHDRØ
          EXEC  CICS WRITEQ TD                                       X
                     QUEUE('MXTP')                                   X
                     FROM(TDREC)                                     X
                     LENGTH(TDLEN)                                   X
                     RESP(RESP1)                                     X
                     RESP2(RESP2)
          MVC   TDREC,TDHDR1               INITIALIZE OUTPUT
          MVC   WORKMASK(6),MASK1          EDIT MASK
          ED    WORKMASK(6),GADATE+1       CHARACTERIZE DATE
          MVC   TDREC+17(5),WORKMASK+1     MOVE IN THE DATE
          XR    R4,R4                      CLEAR OUT R4
```

```
          L     R5,GATIME                  GET TIME IN 100THS
          D     R4,=F'360000'              WHOLE HOURS IN R5
          CVD   R5,WORKTIME                MAKE HOURS PD
          MVC   WORKMASK,MASK1             EDIT MASK
          ED    WORKMASK,WORKTIME+5
          MVC   TDREC+26(2),WORKMASK+4     MOVE HOURS TO HEADER
          LR    R5,R4                      MOVE REMAINDER IN
          XR    R4,R4                      CLEAR OUT R4
          D     R4,=F'6000'                WHOLE MINUTES IN R5
          CVD   R5,WORKTIME                MAKE MINUTES PD
          MVC   WORKMASK,MASK1             EDIT MASK
          ED    WORKMASK,WORKTIME+5
          MVC   TDREC+29(2),WORKMASK+4     MOVE MINUTES TO HEADER
          LR    R5,R4                      MOVE REMAINDER IN
          XR    R4,R4                      CLEAR OUT R4
          D     R4,=F'100'                 WHOLE SECONDS IN R5
          CVD   R5,WORKTIME                MAKE SECONDS PD
          MVC   WORKMASK,MASK1             MOVE IN MASK FOR EDIT
          ED    WORKMASK,WORKTIME+5
          MVC   TDREC+32(2),WORKMASK+4     MOVE SECONDS TO HEADER
* WRITE OUT TIME/DATE OF MXT EVENT
          EXEC  CICS WRITEQ TD                                        X
                     QUEUE('MXTP')                                    X
                     FROM(TDREC)                                      X
                     LENGTH(TDLEN)                                    X
                     RESP(RESP1)                                      X
                     RESP2(RESP2)
* WRITE OUT COLUMN HEADINGS
          MVC   TDREC,TDHDR2
          EXEC  CICS WRITEQ TD                                        X
                     QUEUE('MXTP')                                    X
                     FROM(TDREC)                                      X
                     LENGTH(TDLEN)                                    X
                     RESP(RESP1)                                      X
                     RESP2(RESP2)
          LA    R4,GATRANS
          LA    R5,GATRANS+2800
LOOP2     EQU   *
          MVC   TDREC,TDDET1
          MVC   WORKMASK,MASK1
          ED    WORKMASK,1(R4)             TASK NUMBER
          MVC   TDREC(5),WORKMASK+1
          MVC   TDREC+6(4),4(R4)           TRANSACTION ID
          MVC   TDREC+11(8),16(R4)         RESOURCE TYPE
          MVC   TDREC+20(8),8(R4)          RESOURCE NAME
          MVC   TDREC+29(4),24(R4)         TERMINAL ID
          CLI   0(R4),X'99'                CURRENTLY DISPATCHED TASK?
          BNE   WRTLINE                    NO - SKIP HILITE
          MVC   TDREC+34(4),=C'****'       HILITE THIS TASK ON OUTPUT
WRTLINE   EQU   *
```

```
* WRITE OUT TRANSACTION INFO (FOR EACH TRANSACTION)
          EXEC  CICS WRITEQ TD                                          X
                     QUEUE('MXTP')                                      X
                     FROM(TDREC)                                        X
                     LENGTH(TDLEN)                                      X
                     RESP(RESP1)                                        X
                     RESP2(RESP2)
          LA    R4,28(,R4)                NEXT ENTRY
          CR    R4,R5                     END OF ARRAY
          BH    NEXT1                     YES - PROCESS NEXT SLOT
          CLC   Ø(4,R4),=C'    '          ANY MORE DATA?
          BNE   LOOP2                     YES - GO AND WRITE IT
* TO FINISH OFF, WRITE A BLANK SEPARATION LINE
          MVC   TDREC,TDHDRØ
          EXEC  CICS WRITEQ TD                                          X
                     QUEUE('MXTP')                                      X
                     FROM(TDREC)                                        X
                     LENGTH(TDLEN)                                      X
                     RESP(RESP1)                                        X
                     RESP2(RESP2)
*-------------------------------------------------------------------*
* CHECK FOR ANY MORE EVENTS WAITING FOR PRINTING                     *
*-------------------------------------------------------------------*
NEXT1    EQU   *
          LA    R8,28Ø8(,R8)              NEXT SLOT ADDR
          LA    R6,GALMENT                ADDRESS OF ARRAY
          L     R4,=F'28Ø8Ø'              LENGTH OF SLOT
          AR    R6,R4                     ADD TOGETHER
          CR    R8,R6                     END OF ARRAY?
          BL    LOOP1                     NO - CHECK NEXT ONE
*-------------------------------------------------------------------*
* UPDATE LAST PRINTED DATE/TIME, CANCEL ANY OTHER ZMPR TRANSACTION   *
*  WHICH MAY BE WAITING THEN RE-START ZMPR IN 3 MINS TIME.           *
*-------------------------------------------------------------------*
RETURN   EQU   *
          MVC   GALPRT,CDATE              UPDATE LAST PRINT DATE
          EXEC  CICS CANCEL                                          X
                REQID('CZSMXTPR')                                    X
                RESP(RESP1)
          EXEC  CICS START TRANSID('ZMPR')                           X
                AFTER MINUTES(3)                                     X
                REQID('CZSMXTPR')                                    X
                RESP(RESP1)
          EXEC  CICS RETURN
*-------------------------------------------------------------------*
* WRITE ERROR MSG AND RETURN                                         *
*-------------------------------------------------------------------*
ERRORGA1 EQU   *
          MVC   TDREC,ERRMSG1
          EXEC  CICS WRITEQ TD                                       X
```

```
                    QUEUE('MXTP')                                       X
                    FROM(TDREC)                                         X
                    LENGTH(TDLEN)                                       X
                    RESP(RESP1)                                         X
                    RESP2(RESP2)
          B     RETURN
          LTORG
LABEL1    DC    CL24'***CZSMXTRP DFHEISTG ***'
TDHDRØ    DC    CL133' '
TDHDR1    DC    CL133'MXT SUMMARY FROM YYDDD AT HH:MM:SS'
TDHDR2    DC    CL133'TASK# TRAN TYPE     NAME     TERMID'
TDDET1    DC    CL133'NNNNN NNNN NNNNNNNN NNNNNNNN NNNN'
ERRMSG1   DC    CL133'EXTRACT EXIT COMMAND FAILED'
MASK1     DC    X'FØ2020202020'
          END
```

Peter Duvall
Designer/Developer
Cooperative Bank (UK)

CICS questions and answers

Q Can you recommend a method to delete unused TSQs? We have been monitoring our TS usage because we are about to move TS to the Coupler and have concerns with the build up of 'orphan' TSQs.

A There are a few methods that can be deployed to manage TSQs. The LASTUSEDINT value of the INQUIRE and SET TSQUEUE/TSQNAME commands can be queried to determine the last time a TSQ was used. A program could be written to run every hour (or other interval) and delete TSQs that have not been referenced for a certain time period. Logic needs to be added to exclude IBM and vendor product TSQs and any TSQs needed by your application that are required but not often referenced. CICS internal queues are prefixed: '**', '$$', X'FA' through to X'FF', CEBR, and DF.

The terminal auto-install program can also be modified (terminal delete logic) to delete TSQs relating to the terminal

with which the user has just disconnected from CICS. Often these TSQs are either prefixed or suffixed with the CICS TERMID.

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

Gluecode Software has announced the availability of its Enterprise Server 3.5, its first enterprise-class Open Source business automation server.

The product combines business process management, security management, and an enterprise portal as an integrated suite. With this customers can build customized applications that streamline business processes and information delivery, leading to increased productivity and response times within and across organizations – they claim.

The Server broadens the access to enterprise applications with built-in JCA connectors, including SAP, Siebel, PeopleSoft, JD Edwards, MQSeries, and CICS.

In addition, the product supports Apache Cocoon, which allows centralized management of disparate information sources, which in turn enables companies to pipeline data from multiple sources, and automatically publish the information in a meaningful and actionable format for the end-user.

For further information contact:
Gluecode Software, 2321 Rosecrans Avenue, Suite 2205, El Segundo, CA 90245, USA.
Tel: (310) 536 8355.
URL: http://www.gluecode.com/website/html/news_pres_relpg3.htm.

* * *

Micro Focus and Microsoft have announced a new alliance to enable the migration of critical proprietary mainframe systems onto Windows using Microsoft .NET technology.

The alliance lays the technology foundation to move application workloads from the mainframe to Intel architecture and the Windows Server platform. It claims to help customers reduce the cost of maintaining and modernizing their mainframe environments, saying that cheaper platforms save time and money.

Micro Focus Enterprise Server with its new Mainframe Transaction Option underpins the new platform alliance and now enables the migration and deployment of CICS/COBOL mainframe applications to the Windows platform. Enterprise Server offers a way of re-hosting mainframe applications on Windows. Once the application has been re-hosted, it can be extended through the use of the .NET Framework, SQL Server 2000, XML, and Web services.

For further information contact:
Micro Focus, 9420 Key West Avenue, Rockville, MD 20850, USA.
Tel: 301 838 5000.
URL: http://www.microfocus.com/press/news/20040413b.asp.

* * *

IBM has announced CICS VSAM Transparency for z/OS V1.1, which enables the migration of data from VSAM to DB2. It ensures continued access to that data in DB2, without modification to CICS or batch VSAM application programs and with only minor changes to CICS and batch configuration.

Whether for statutory compliance requirements or the need to modernize CICS applications, CICS VSAM Transparency for z/OS can help customers unlock value in data used by their legacy applications and extend its use to new DB2 applications.

For further information contact your local IBM representative.
URL: http://www.ibm.com/software/htp/cics/vt.