CICS update

237

August 2005

In this issue

A tcl PUBLICATION

CICS Update

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

Colin Smith
E-mail: info@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.

CICS TS 3.1 Web Services Assistant

A major step towards interoperability is taken in the latest release of CICS TS (Version 3.1), which allows CICS applications to be well integrated with the Service-Oriented Architecture (SOA). In other words, now your CICS application can act either as a Web service provider or a Web service requestor. This opens a whole new set of ways in which existing CICS applications can be exploited in an enterprise.

Obviously there are multiple ways in which a wrappering solution can be implemented. Using the IBM-supplied utilities, called Web Services Assistant, is one way that this can be achieved with relative ease.

BASIC COMPONENTS OF CICS WEB SERVICES

Before we proceed with how to enable CICS as a Web service, let us establish some of the basic requirements of CICS Web services and how it functions.

The execution environment that allows a particular CICS application program to operate in a Web services setting is defined by three objects – the pipeline, the Web service binding file, and the Web service description. These objects are defined to CICS as attributes of the WEBSERVICE resource definition.

MESSAGE HANDLERS AND PIPELINES

When CICS is the service provider, it basically receives the request from the service requestor, examines it, and extracts the relevant contents, and with this data invokes the target application program. Once the application program returns control back to CICS, it constructs the response using the data returned by the program and sends it back to the service requestor.

When CICS processes a service request, it uses a pipeline, which is just a set of message handlers. The message handler is a program that performs its own processing of Web service requests and responses. For example, there could be a message handler for encryption/decryption, audit logging, security checking etc. The last handler in the pipeline – also called the terminal handler – is the one that invokes the target application program. The number of message handlers in a pipeline is configurable.

Each message handler typically processes the message and passes it on (either modified or unmodified) to the next handler. But occasionally, the message handler itself can short-circuit the process and send back a response (eg a security failure).

If we assume that there is a pipeline with three message handlers (each passing the message to the next one after doing their job), then, during the request phase, CICS passes control from handler 1 to 2 to 3 and then to the target application program. On return from the application program, CICS invokes the handlers in the reverse order (ie 3, 2, and 1 respectively) before passing the response to the service requestor.

CICS, by default, provides SOAP message handlers and associated pipeline configuration files like:

- basicsoap11provider.xml – a pipeline configuration file for a service provider using the SOAP 1.1 message handler.
- basicsoap11requester.xml – a pipeline configuration file for a service requester using the SOAP 1.1 message handler.

WEB SERVICE BINDING FILE

The Web service binding file contains information that CICS uses to map data between input and output messages, and application data structures. CICS uses this information at run-time to map the application data structure and SOAP messages. Web service binding files are defined to CICS in the WSBIND attribute of the WEBSERVICE resource.

WEB SERVICE DEFINITION

A Web service description (WSDL) contains abstract representations of the input and output messages used by the Web service. The Web services definition either can be an existing one or may be generated from an application language data structure using the utility program DFHLS2WS. CICS uses the Web service description to perform full validation of SOAP messages. At run-time, CICS performs the mapping between the application data structures and the messages.

With new applications, it is recommended that the business logic be separated from the Web services logic using wrapper programs.

NEW CICS RESOURCES

Web services in CICS uses three new CICS resources – PIPELINE, URIMAP, and WEBSERVICE.

PIPELINE

A PIPELINE resource definition specifies the information about the message handlers to be applied to a Web service request.

The pipeline configuration file is an XML document (HFS file) used to contain the XML description of the nodes and their configuration. The name of the file is specified in the CONFIGFILE attribute of a PIPELINE definition.

The WSDIR attribute of the PIPELINE definition can be used to specify the valid HFS directory containing the Web services binding files associated with the pipeline. If specified when the PIPELINE definition is installed, CICS uses the scanning mechanism and automatically installs the Web services binding files from this directory.

Typically, many applications can use the same PIPELINE definition, and hence a pipeline need not be defined for each service.

WEBSERVICE

A WEBSERVICE resource defines the pipeline, the Web service binding file, and the Web service description aspects of the run-time environment for a CICS application program deployed in a Web services setting. This is used where the mapping between the application data structure and SOAP messages has been generated using the CICS Web Services Assistant.

Using the PERFORM PIPELINE command, the CICS scanning mechanism can be used to dynamically install WEBSERVICE resources. The advantages of this approach are that it reduces the amount of resource definition required, and CICS can make direct use of information provided at development time.

URIMAP

URIMAP definitions enable CICS to match the URIs of requests from Web clients, or requests to a remote server, and provide information on how to process the requests. URIMAP definitions for Web service requests have a USAGE attribute of PIPELINE. These URIMAP definitions associate a URI for an inbound Web service request with a PIPELINE or WEBSERVICE resource that specifies the processing to be performed.

WHAT IS CICS WEB SERVICES ASSISTANT?

CICS Web Services Assistant is a set of batch utilities that support deployment of CICS applications as service providers or service requestors – with minimal programming effort.

Basically it has the following features:

- It can create a WSDL document from a simple language structure (so that an existing CICS application can be exposed as a Service – CICS acts as a service provider) and similarly create a language structure from an existing WSDL document (so that CICS can exploit an existing Web service – CICS as a service requestor). COBOL, C, C++, and PL/I are the languages supported.

- It can generate information required to enable automatic run-time conversion of the SOAP messages to application programs' data structures and *vice versa*.

The CICS Web Services Assistant comprises two utility programs:

1. DFHLS2WS, which generates a Web service binding file from a language structure. It also generates a Web service description.
2. DFHWS2LS, which generates a Web service binding file from a Web service description. It also generates a language structure that can be used in application programs.

The JCL procedures to run both these programs are in the *hlq*.XDFHINST library.

SIMPLE DEPLOYMENT OF AN EXISTING CICS APPLICATION AS A SERVICE PROVIDER

Let us make the following assumptions, so that we have a simple scenario to understand the process of enabling an existing CICS application as a service provider:

1. The existing CICS application is well-structured, in the sense that the communication layer and the business logic are separated – this enables the CICS Web Assistant to connect directly to the business logic.
2. The Web service to be created is a new one – this implies that the data structure in the existing program can be the driving factor.
3. The existing program is in COBOL and the input and output data structures are the same.

4 The data structures are defined separately in a copybook.

5 SOAP is the message format selected.

6 The CICS-supplied pipeline configuration file is sufficient.

The following is the process to be adopted:

1 With the COBOL copybook as input, create the Web services binding file by using the batch utility DFHLS2WS.

2 Copy the Web service binding file (WSBIND) to the pick-up directory of the PIPELINE resource to be used for your Web service application.

3 Create the URIMAP – which matches the URI used to invoke the Web service – automatically from the Web services binding file using the scan mechanism. Install the URIMAP. The URIMAP specifies the names of the WEBSERVICE resource and the PIPELINE resource that provide further details of how the Web service request is processed.

4 Create the WEBSERVICE – which specifies the location of the WSDL and the WSBIND file – by scanning the WSBIND files. Install the WEBSERVICE that is consistent with the WSDL.

Now you have set up your application program to be a service provider, and, by publishing the Web service description, service requestors can make use of your new Web service.

WHEN TO WRITE A WRAPPER PROGRAM

When the CICS Web Services Assistant cannot generate code that can directly interface with your target application business logic, you can use a wrapper program, which the CICS Web services support would interface with.

The reason for doing this could be that the data structure used by your business logic cannot be directly mapped to the SOAP message (either because of incompatible data types in your application program or because of the need to meet the existing Web services definition, which is different from what your application program can offer). In such cases, the wrapper program can just handle this interfacing with the existing target application program and appropriately move data from these two different data structures.

CONCLUSION

The option of opening up existing CICS applications as Web services and also enabling CICS programs to integrate with Web services in other distributed environments is clearly a major enabler in adapting Service-Oriented Architecture in large enterprises that have their core business in CICS-based applications.

As described in this article, exploiting this could be fairly simple, provided the existing applications are fairly well-structured and lend themselves to being exposed as services. But this is rarely the case because most of these are age-old applications and were never intended to be used as Web services. Extensive planning is recommended as to which of the applications are to be exposed, and a strategy needs to be arrived at for using the most appropriate techniques for achieving this.

Sasirekha Cota
System Software Group
Tata Consultancy Services (India)

Temporary storage analysis and tuning

CICS produces statistical information about the various resources managed within the system. This article examines the statistics generated by the temporary storage domain. It highlights a number of attributes that are worth analysing in detail when reviewing temporary storage usage and performance in a CICS system. It also offers guidance on changes that may be made to the temporary storage environment in order to improve system performance, reduce I/O activity, and increase transaction throughput.

STATISTICS RECAP

CICS generates statistics for analysis of its resources. Statistical information is recorded by CICS at various times: periodically at user-specified time intervals, at CICS end-of-day processing, during shutdown of the CICS system, and also dynamically using the supplied sample transaction STAT. This runs the supplied sample COBOL program DFH0STAT. There is also the CICS utility program DFHSTUP, which formats CICS statistics information offline.

CICS domains generate statistical information to contribute towards the overall CICS system statistics. The temporary storage domain is one such example. In the case of temporary storage, the domain statistics are also formatted when running the CICS system dump formatting VERBX on the temporary storage domain. In the case of CICS TS 2.3 (for example), the appropriate VERBX is DFHPD630. Using the Interactive Problem Control System (IPCS), and formatting a CICS system dump with the VERBX DFHPD630 formatting option, specifying 'TS' as the domain to be analysed, the formatted dump output contains statistical information at the start of the TS dump summary section.

ANALYSING AN EXAMPLE SET OF TEMPORARY STORAGE STATISTICS

Initially, an example set of statistical data from a sample CICS system will be analysed:

```
Put/Putq main storage requests . . . . . :     943,819
Get/Getq main storage requests . . . . . :   5,632,822
Peak storage used for TS Main. . . . . . :       5,981K
Current storage used for TS Main . . . . :       5,653K
Put/Putq auxiliary storage requests. . . :     351,811
Get/Getq auxiliary storage requests. . . :     589,542
```

```
Times temporary storage queue created. . :     393,121
Peak temporary storage queues in use . . :      19,611
Current temporary storage queues in use. :      19,313
Items in longest queue . . . . . . . . . :       1,310
Control interval size. . . . . . . . . . :       4,096
Control intervals in the DFHTEMP dataset :      65,535
Peak control intervals used. . . . . . . :      32,967
Current control intervals in use . . . . :      32,948
Available bytes per control interval . . :       4,032
Segments per control interval. . . . . . :          63
Bytes per segment. . . . . . . . . . . . :          64
Writes bigger than control interval size :      25,711
Largest record length written. . . . . . :      15,914
Times auxiliary storage exhausted. . . . :           0
Number Temporary storage compressions. . :      46,911
Temporary storage strings. . . . . . . . :          60
Peak Temporary storage strings in use. . :          14
Temporary storage string waits . . . . . :           0
Peak users waiting on string . . . . . . :           0
Current users waiting on string. . . . . :           0
Temporary storage buffers. . . . . . . . :         360
Temporary storage buffer waits . . . . . :         348
Peak users waiting on buffer . . . . . . :           1
Current users waiting on buffer. . . . . :           0
Temporary storage buffer reads . . . . . :     298,121
Temporary storage buffer writes. . . . . :      14,003
Forced buffer writes for recovery. . . . :           0
Format writes. . . . . . . . . . . . . . :           0
I/O errors on the DFHTEMP dataset. . . . :           0
Shared Pools defined . . . . . . . . . . :           0
Shared Pools currently connected . . . . :           0
Shared temporary storage read requests . :           0
Shared temporary storage write requests. :           0
```

OBSERVATIONS ABOUT THE DATA

These statistics show a reasonably healthy CICS system usage of temporary storage, although (as we shall see) things could be improved by a simple adjustment to one of the parameters.

The statistics show that there were 943,819 writes to main temporary storage, and 5,632,822 reads of this main temporary storage data. Peak storage usage for main temporary storage data was just below 6MB. The system was also a busy user of auxiliary temporary storage. There were 351,811 writes to auxiliary temporary storage queues, and 589,542 reads of this data back. The statistics show that almost 400,000 queues were created, and that the peak number of queues in use was 19,611. When the data was gathered, 19,313 queues were in use. This indicates that the peak queue usage is comparable to the current number of queues in the system. This could be interpreted as meaning that the system is close to a state of equilibrium with regards to queue creation and deletion, deleting queues as soon as they are no longer required. The biggest queue had 1,310 records on it; the largest record written to a queue was 15,914 bytes long.

The data regarding the auxiliary queue management is as follows. The control interval (CI) size on DFHTEMP is 4,096 bytes. Note that CICS reserves a small part of each CI for its own use – this means that at best a CI can never have all its space available for temporary storage data. This is why the *Available bytes per control interval* value of 4,032 is less than the *Control interval size*.

Of the 65,535 CIs on DFHTEMP, 32,948 were currently in use, from a peak usage of 32,967. There had been 25,711 writes of data that exceeded the CI size. Temporary storage buffer compressions (to free up unused space scattered within CIs in buffers) occurred 46,911 times. There were 60 strings available for VSAM I/O to DFHTEMP, and there had been a peak of 14 strings in use. In addition, there were 360 temporary storage buffers in the CICS address space, and there had been 348 task waits for a buffer to be freed for use (no more than one task had ever waited for a buffer at any one time). CICS had read CI data into buffers 298,121 times, and had flushed buffers to DFHTEMP 14,003 times. No formatting writes to DFHTEMP, and no I/O errors, had occurred.

The statistics also showed that shared temporary storage was not in use by this CICS system.

Given the above breakdown of temporary storage usage, the question to be asked is, how could the results be improved upon? Any improvement should aim to increase the positive aspects shown by the statistics, and decrease the negative ones. Negative aspects include the fact that 348 task waits for a buffer occurred. Analysis also shows that a large amount of I/O activity was taking place to write (and particularly to read) CI data to and from buffers. A reduction in this I/O activity would be a benefit to system throughput and performance. In addition, the relatively large number of data items written that were bigger than the CI size indicates that an overhead of additional CICS processing was required to accommodate these large records. If this number of records larger than a CI could be reduced, it would also be beneficial to the system.

A SIMPLE CHANGE

It is a fact that the size of a CI has a direct bearing on the temporary storage I/O activity and the number of records written that exceed the CI size. A smaller CI size will result in fewer records that can fit into a CI. This means that CIs will fill up faster (all things being equal). It also means that CICS will have to flush buffers more often in order to move temporary storage data records out of buffers in memory and onto DFHTEMP, and in doing so allow the buffer to be allocated to a new (empty) CI to hold new record data. In addition to this, a future read of data will be less likely to find the target record it wants in a buffer in core, and so more likely to need temporary storage processing to flush a buffer so that the actual CI containing the target record can be read in. Also, writes will have to do a similar operation in order to place a CI with sufficient space into a buffer, so that the record being written can be accommodated.

Increasing the CI size provides several benefits. The flushing of temporary storage buffers should be reduced. This is because larger CIs provide more space for records to be written into before an I/O is required. Similarly, they are more likely to hold a target record being read, and so reduce the likelihood of buffer flushes and DASD reads in order to get the required CI containing the record into the address space, so that it may be returned to the application. Again, if a sufficient number of CIs can be held in core in temporary storage buffers, and if the buffers (CI sizes) are big enough, the need to read in different CIs with enough space to hold data being written is reduced.

A downside with bigger CIs is the extra cost of their I/O, but with modern DASD this is less of an issue than it used to be. There is also the potential for buffer compressions to have to perform additional work (because of the additional number of records held within the larger CIs). Another downside is the increased usage of extended CICS dynamic storage (ECDSA storage) to accommodate the larger buffers, but with sensible workload balancing, function shipping, and the use of Queue Owning Regions (QORs), this is not unreasonable to achieve.

If a CICS system is not constrained for virtual storage, one simple way of improving performance can be to increase the CI size of the CIs on DFHTEMP. If the number of buffers is maintained at the same value (assuming the extra size of the CIs can be accommodated in terms of the extra size of the buffers needed to hold them in core) this change can provide a number of benefits.

NEW STATISTICS

The results of changing the CI size from 4,096 to 8,192 are shown below. This is the only change made to the settings for this example CICS system.

```
Put/Putq main storage requests . . . . . :     891,855
Get/Getq main storage requests . . . . . :   3,812,511
Peak storage used for TS Main. . . . . . :       5,914K
Current storage used for TS Main . . . . :       5,9Ø5K
Put/Putq auxiliary storage requests. . . :     358,993
Get/Getq auxiliary storage requests. . . :     3Ø6,5Ø1
Times temporary storage queue created. . :     392,993
Peak temporary storage queues in use . . :       6,712
Current temporary storage queues in use. :       6,415
Items in longest queue . . . . . . . . . :       1,Ø66
Control interval size. . . . . . . . . . :       8,192
Control intervals in the DFHTEMP dataset :      27,ØØ4
Peak control intervals used. . . . . . . :       4,843
Current control intervals in use . . . . :       4,786
```

```
Available bytes per control interval . . :       8,128
Segments per control interval. . . . . . :         127
Bytes per segment. . . . . . . . . . . . :          64
Writes bigger than control interval size :         381
Largest record length written. . . . . . :      15,914
Times auxiliary storage exhausted. . . . :           Ø
Number Temporary storage compressions. . :      5Ø,512
Temporary storage strings. . . . . . . . :          6Ø
Peak Temporary storage strings in use. . :           4
Temporary storage string waits . . . . . :           Ø
Peak users waiting on string . . . . . . :           Ø
Current users waiting on string. . . . . :           Ø
Temporary storage buffers. . . . . . . . :         36Ø
Temporary storage buffer waits . . . . . :           Ø
Peak users waiting on buffer . . . . . . :           Ø
Current users waiting on buffer. . . . . :           Ø
Temporary storage buffer reads . . . . . :      58,543
Temporary storage buffer writes. . . . . :       2,718
Forced buffer writes for recovery. . . . :           Ø
Format writes. . . . . . . . . . . . . . :           Ø
I/O errors on the DFHTEMP dataset. . . . :           Ø
Shared Pools defined . . . . . . . . . . :           Ø
Shared Pools currently connected . . . . :           Ø
Shared temporary storage read requests . :           Ø
Shared temporary storage write requests. :           Ø
```

The new set of statistics shows a number of improvements to the system. First, it should be noted that the temporary storage system activity and throughput at the time the new statistics were taken is comparable to that when the original statistics were gathered. This is good, since it means that a sensible comparison of the two sets of data may be performed.

With the larger CI size, there are now about half the number of CIs in the DFHTEMP dataset (27,004 from 65,535). The peak number of CIs in use has reduced to a fraction of its previous value (4,843 from 32,967). Similarly, the current number of CIs in use has reduced too (4,786 from 32,948). These figures indicate that the larger CIs are big enough to hold temporary storage record data for its normal duration, that is, the data is deleted before new temporary storage requests come along requiring the use of additional CIs. By having larger CIs that can therefore hold more record data there is a reduction in the need to use more CIs while data is in the system. This is beneficial from an I/O perspective too.

The number of writes larger than the CI size has reduced considerably (381 from 25,711). This avoids the overhead inherent in CICS having to divide record data across multiple CIs, and in turn reduces CPU usage within the temporary storage domain.

The number of buffer compressions has remained fairly static (50,512 from 46,911). In part, this may be explained by the comparable workloads and the fact that the same volume of data is being referenced in the two runs, leading to a similar number of buffer selections and compressions.

There are now no tasks being made to wait for buffer availability (0 from 348). This is good news from the perspective of CICS transaction throughput. Perhaps best of all, the DFHTEMP I/O activity has reduced considerably. Buffer reads (I/O to place a CI into a buffer) have reduced (58,543 from 298,121) as have buffer writes (to flush a CI to DFHTEMP and so release a buffer for reuse), which have reduced from 14,003 to 2,718.

Taking this strategy further, one could define more buffers, or increase their CI size further. Although it would result in additional virtual storage requirements for temporary storage management, having sufficient buffers for all CIs that were required at any point in time would mean that I/O was never required by temporary storage to flush out one CI to disk in order to free up buffer space for a required CI to be read in to. This would make auxiliary temporary storage very efficient. Indeed, by having pre-allocated buffer storage (so avoiding the need for GETMAINs), coupled with no I/O, auxiliary temporary storage can outperform main temporary storage!

Note that the CICS system initialization parameter 'TS=' controls the number of buffers and strings, with default values of 3 and 3. The theoretical maximum number of auxiliary storage buffers is 32,767.

FURTHER READING

For additional details of the various pieces of state data recorded by the temporary storage domain, readers are referred to the *CICS Performance Guide*. This also gives details on the lifetime of the data (ie when/if the values are reset).

CONCLUSION

I hope that this article has helped explain the background to interpretation of temporary storage statistics, and their use in tuning CICS systems.

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

Exits XZCIN and XZCOUT to translate ASCII/ EBCDIC

In many installations it is necessary to have Automatic Teller Machines (ATMs) with a direct connection to the host. These ATMs can, for example, use a communication protocol like X.25 non-SNA, and are supported by CICS as an LUTYPE 1.

Because the protocol is non-SNA, the ATM does not have a way to convert ASCII characters to EBDIC for CICS to receive, nor is there a way to convert the EBCDIC characters to ASCII when CICS gives back the answers to the ATMs.

The solution is to code two Assembler CICS exits:

- XZCIN, which allows CICS to receive the messages sent to it by the ATM by translating them from ASCII to EBCDIC.
- XZCOUT, which allows the ATM to receive the messages sent to it by CICS and translate them from EBCDIC to ASCII.

These exits have been shown to work in an environment running z/OS 1.4 and CICS TS.1.3, and even CICS TS 2,2, and in both cases they use the TCTTE and the TIOA as input parameters, and the output is the translated TIOA.

It has also been tested on CICS/ESA 4.1.0 and OS/390 2.10.

Both exits must be link-edited with AMODE=31 and RMODE=ANY.

The tables of ASCII/EBCDIC conversion can be personalized according to the requirements of your installation.

The following is a brief description of what the exits XZCIN and XZCOUT do:

- The exit recovers the TIOA and the TCTTE. It compares the type of terminal (TCTTETT) with the values defined in CICS for that type of LU. In our case it’s 3767, and the possible values are X'BE' or X'BC'. If the type of terminal is the same, it continues with the comparison and the terminal is identified (TCTTETI).
- If the name of the terminal in CICS begins with ‘AT’ (test environment) it goes to the label ‘SIGUE’, otherwise it looks for a terminal beginning with ‘U’ (production environment). If this test is satisfied, it branches off to label ‘SIGUE’, like in the test environment.

The routine ‘SIGUE’ translates the content of the TIOA, byte-by-byte, from ASCII to EBCDIC in the case of XZCIN and from EBCDIC to ASCII in the case of the XZCOUT.

The exits can be activated by a program that is included in the PLTPI and executed in CICS start-up, for example:

```
ENA-XZCIN.

EXEC CICS INABLE PROGRAM('XZCIN') EXIT (' XZCIN') START
                           END-EXEC.

ENA-XZCOUT.
EXEC CICS INABLE PROGRAM('XZCOUT') EXIT (' XZCOUT') START
                           END-EXEC.
```

XZCIN ASS

```
         TITLE 'input event exit ***  XZCIN  '
         PRINT   NOGEN
***********************************************************************
*        Sample user exit program for task attach (XZCIN )            *
*                                                                     *
* STATUS = 4.1.Ø                                                      *
*                                                                     *
* FUNCTION =                                                          *
*        This is a sample user exit program to be invoked at the      *
*        XZCIN  global user exit point when processing Input Event.   *
*                                                                     *
*        It translates the Terminal Input Output area from ASCII to   *
*        EBCDIC if the terminal is an Automatic Teller Machine and    *
*        your terminal id begins with 'AT' or 'U'.                    *
*                                                                     *
* NOTES :                                                             *
*    DEPENDENCIES = S/37Ø                                             *
*    RESTRICTIONS = None                                              *
*    PATCH LABEL = None                                               *
*    MODULE TYPE = Executable                                         *
*    PROCESSOR = Assembler                                            *
*    ATTRIBUTES = Read only, Serially Reusable                        *
*                                                                     *
* AUTHOR : Hernan Zamudio Enzian.                                     *
*                                                                     *
* MACROS = DFHUEXIT TYPE=EP,ID=(XZCIN )                               *
*         Generates the User Exit Parameter list for the XZCIN        *
*         global user exit point.                                     *
***********************************************************************
         SPACE
RØ       EQU   Ø                          NOT USED
R1       EQU   1                          INITIAL USER EXIT PARAMETER LIST
R2       EQU   2                          USER EXIT PARAMETER LIST
R3       EQU   3                          NOT USED
R4       EQU   4                          NOT USED
R5       EQU   5                          CICS SHARED STG BELOW 16MB (64K)
R6       EQU   6                          CICS SHARED STG ABOVE 16MB (128K)
R7       EQU   7                          NOT USED
R8       EQU   8                          TIOA ADDRESS
R9       EQU   9                          NOT USED
R1Ø      EQU   1Ø                         TCTTE ADDRESS
R11      EQU   11                         NOT USED
R12      EQU   12                         PROGRAM BASE
R13      EQU   13                         SAVE AREA
R14      EQU   14                         RETURN ADDRESS
R15      EQU   15                         INITIAL PROGRAM BASE
         EJECT
         DFHUEXIT TYPE=EP,ID=(XZCIN)
         EJECT
         PRINT   ON
         COPY  DFHTCTTE
         PRINT   NOGEN
TCTTEAR  EQU   R1Ø
         PRINT ON
         COPY  DFHTIOA
TIOABAR  EQU   R8
DFH$XCIN CSECT
DFH$XCIN AMODE 31
DFH$XCIN RMODE ANY
         SAVE  (14,12)            SAVE REGS
         LR    R11,R15            SET-UP BASE REGISTER
         USING DFH$XCIN,R11       ADDRESSABILITY
         LR    R2,R1              GET UEP PARAMETER LIST
         USING DFHUEPAR,R2        ADDRESSABILITY
         SPACE
         L     R1Ø,UEPTCTTE       ADDRESSABILITY TCTTE
         L     R8,UEPTIOA         ADDRESSABILITY TIOA
         LA    R3,TIOADBA         ADDRESABILITY  TIOADBA
         LH    R6,TIOATDL
COMPARA  CLI   TCTTETT,X'BD'      Contention Logical Unit ?
         BE    COMPARA2
COMPARA1 CLI   TCTTETT,X'BE'      Interactive Logical Unit ?
         BNE   EXIT
COMPARA2 CLC   TCTTETI(2),=C'AT'  It's an ATM Development ?
         BE    SIGUE              NO
         CLI   TCTTETI,C'U'       It's an ATM Production ?
         BNE   EXIT               NO
SIGUE    DS    ØH
         CH    R6,=H'5'           TIOA Length > 5 ?
         BH    LOOP               YES
         B     EXIT               NO, Exit
LOOP     DS    ØH
         MVC   WKB,Ø(R3)
ASCEBC   DS    ØH
         TR    WKB,IECTRASA
MOVE     DS    ØH
         MVC   Ø(1,R3),WKB
         LA    R3,1(R3)
         BCT   R6,LOOP
         B     EXIT
EXIT     DS    ØH                 RETURN TO THE CALLER
         L     R13,UEPEPSA        ADDRESS OF EXIT SAVE AREA
         RETURN (14,12),RC=UERCNORM    RESTORE REGS AND RETURN
WKB      DC    X'ØØ'              WORK AREA
***********************************************************************
* * *               T R A N S L A T I O N   T A B L E S               *
***********************************************************************
IECTSASA EQU   *                  USASCII OUTPUT TRANSLATE TABLE
*                 Ø 1 2 3 4 5 6 7 8 9 A B C D E F
```

```
         DC    X'ØØØ1Ø2Ø3BØØ9B17FØ8B3B4ØBØCØDØEØF' Ø
         DC    X'1Ø111213B9ØAØ8BB1819BCBF1C1D1E1F' 1
         DC    X'C2C31CC4C5ØA171BC8C9CACBCCØ5Ø6Ø7' 2
         DC    X'CDCE1691921E9FØ4D9DADBDC1415DD1A' 3
         DC    X'2ØDF838485AØC686877C5B2E3C282B21' 4
         DC    X'268288898AA18C8B8DE15D242A293B5E' 5
         DC    X'2D2FB68EB7B5C78F8Ø237C2C255F3E3F' 6
         DC    X'9B9ØD2D3D4D6D7D8DE6Ø3A234Ø273D22' 7
         DC    X'9D616263646566676869AEAFDØECE7F1' 8
         DC    X'F86A6B6C6D6E6F7Ø7172A6A7EØF7E2CF' 9
         DC    X'E67E737475767778797AE5A8D1EDE8A9' A
         DC    X'BD9CBEFAB8F5F4ACABF35E21AD7EF99E' B
         DC    X'7B414243444546474849FØ939495A2E4' C
         DC    X'7D4A4B4C4D4E4F5Ø5152FB968197A398' D
         DC    X'5CF6535455565758595AFDFFFFE3EEEF' E
         DC    X'3Ø313233343536373839FCEA9AEBE9FF' F
IECTRASA EQU   *                   USASCII INPUT TRANSLATE TABLE
*                 Ø 1 2 3 4 5 6 7 8 9 A B C D E F
         DC    X'ØØØ1Ø2Ø3372D2E2FØ8Ø515ØBØCØDØEØF' Ø
         DC    X'1Ø1112133C3D322618193F271C1D1E1F' 1
         DC    X'4Ø4F7F7B5B6C5Ø7D4D5D5C4E6BØ4B61' 2
         DC    X'FØF1F2F3F4F5F6F7F8F97A5E4C7E6E6F' 3
         DC    X'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6' 4
         DC    X'D7D8D9E2E3E4E5E6E7E8E94AEØ5A5F6D' 5
         DC    X'79818283848586878889919293949596' 6
         DC    X'979899A2A3A4A5A6A7A8A9CØ6ADØA1Ø7' 7
         DC    X'68DC5142434447485253545756586367' 8
         DC    X'713334CBCCCDDBDDDFFFFC7ØB18ØBF36' 9
         DC    X'4555CEDE6A7B9A9BABAF5FB8B7BC8A8B' A
         DC    X'Ø4Ø6Ø8Ø9ØA656264B41415171ABØB21B' B
         DC    X'1C1E2Ø21232446628292A2B2C3Ø319F' C
         DC    X'8CAC727374FF75767738393A3B3E7841' D
         DC    X'9C599EEDCFAAAØ8EAEFEFBFD8DADEEEF' E
         DC    X'CA8FFFB9B6B5E19D9ØBEB3DAFAEAFFFF' F
         LTORG
         END   DFH$XCIN
```

XZCOUT ASS

```
         TITLE 'output event exit ***  xzcout '
         PRINT   NOGEN
***********************************************************************
*        Sample user exit program for output event(XZCOUT)           *
*                                                                     *
* STATUS = 4.1.Ø                                                      *
*                                                                     *
* FUNCTION =                                                          *
*        This is a sample user exit program to be invoked at the      *
*        XZCOUT global user exit point when processing Output Event.  *
*                                                                    *
*        It translates the Terminal Input Output area from EBCDIC to *
*        ASCII if the terminal is an Automatic Teller Machine and    *
*        your terminal id begins with 'AT' OR 'U'.                   *
*                                                                    *
* NOTES :                                                            *
*    DEPENDENCIES = S/37Ø                                            *
*    RESTRICTIONS = None                                             *
*    PATCH LABEL = None                                              *
*    MODULE TYPE = Executable                                        *
*    PROCESSOR = Assembler                                           *
*    ATTRIBUTES = Read only, Serially Reusable                       *
*                                                                    *
* AUTHOR : Hernan Zamudio Enzian.                                    *
*                                                                    *
* MACROS = DFHUEXIT TYPE=EP,ID=(XZCOUT)                              *
*         Generates the User Exit Parameter list for the XZCOUT      *
*         global user exit point.                                    *
**********************************************************************
         SPACE
RØ       EQU   Ø                   NOT USED
R1       EQU   1                   INITIAL USER EXIT PARAMETER LIST
R2       EQU   2                   USER EXIT PARAMETER LIST
R3       EQU   3                   NOT USED
R4       EQU   4                   NOT USED
R5       EQU   5                   CICS SHARED STG BELOW 16MB (64K)
R6       EQU   6                   CICS SHARED STG ABOVE 16MB (128K)
R7       EQU   7                   NOT USED
R8       EQU   8                   TIOA ADDRESS
R9       EQU   9                   NOT USED
R1Ø      EQU   1Ø                  TCTTE ADDRESS
R11      EQU   11                  NOT USED
R12      EQU   12                  PROGRAM BASE
R13      EQU   13                  SAVE AREA
R14      EQU   14                  RETURN ADDRESS
R15      EQU   15                  INITIAL PROGRAM BASE
         EJECT
         DFHUEXIT TYPE=EP,ID=(XZCOUT)
         EJECT
         PRINT   ON
         COPY  DFHTCTTE
         PRINT   NOGEN
TCTTEAR  EQU   R1Ø
         PRINT ON
         COPY  DFHTIOA
TIOABAR  EQU   R8
DFH$XCOU CSECT
DFH$XCOU AMODE 31
DFH$XCOU RMODE ANY
         SAVE  (14,12)             SAVE REGS
         LR    R11,R15             SET-UP BASE REGISTER
         USING DFH$XCOU,R11        ADDRESSABILITY
         LR    R2,R1               GET UEP PARAMETER LIST
         USING DFHUEPAR,R2         ADDRESSABILITY
         SPACE
         L     R1Ø,UEPTCTTE        ADDRESSABILITY TCTTE
         L     R8,UEPTIOA          ADDRESSABILITY TIOA
         LA    R3,TIOADBA          ADDRESABILITY  TIOADBA
         LH    R6,TIOATDL
COMPARA  CLI   TCTTETT,X'BD'       Contention Logical Unit ?
         BE    COMPARA2
COMPARA1 CLI   TCTTETT,X'BE'       Interactive Logical Unit ?
         BNE   EXIT
COMPARA2 CLC   TCTTETI(2),=C'AT'   ATM Development ?
         BE    SIGUE               NO
         CLI   TCTTETI,C'U'        ATM Production  ?
         BNE   EXIT                NO
SIGUE    DS    ØH
         CH    R6,=H'5'            TIOA Length > 5 ?
         BH    LOOP                YES
         B     EXIT                NO, Exit
LOOP     DS    ØH
         MVC   WKB,Ø(R3)
EBCASC   DS    ØH
         TR    WKB,IECTSASA
MOVE     DS    ØH
         MVC   Ø(1,R3),WKB
         LA    R3,1(R3)
         BCT   R6,LOOP
         B     EXIT
EXIT     DS    ØH                  RETURN TO THE CALLER
         L     R13,UEPEPSA         ADDRESS OF EXIT SAVE AREA
         RETURN (14,12),RC=UERCNORM    RESTORE REGS AND RETURN
WKB      DC    X'ØØ'               WORK AREA
***********************************************************************
* * *               T R A N S L A T I O N   T A B L E S               *
***********************************************************************
IECTSASA EQU   *                   USASCII OUTPUT TRANSLATE TABLE
*                 Ø 1 2 3 4 5 6 7 8 9 A B C D E F
         DC    X'ØØØ1Ø2Ø3BØØ9B17FØ8B3B4ØBØCØDØEØF' Ø
         DC    X'1Ø111213B9ØAØ8BB1819BCBF1C1D1E1F' 1
         DC    X'C2C31CC4C5ØA171BC8C9CACBCCØ5Ø6Ø7' 2
         DC    X'CDCE1691921E9FØ4D9DADBDC1415DD1A' 3
         DC    X'2ØDF838485AØC686877C5B2E3C282B21' 4
         DC    X'268288898AA18C8B8DE15D242A293B5E' 5
         DC    X'2D2FB68EB7B5C78F8Ø237C2C255F3E3F' 6
         DC    X'9B9ØD2D3D4D6D7D8DE6Ø3A234Ø273D22' 7
         DC    X'9D616263646566676869AEAFDØECE7F1' 8
         DC    X'F86A6B6C6D6E6F7Ø7172A6A7EØF7E2CF' 9
         DC    X'E67E737475767778797AE5A8D1EDE8A9' A
         DC    X'BD9CBEFAB8F5F4ACABF35E21AD7EF99E' B
         DC    X'7B414243444546474849FØ939495A2E4' C
         DC    X'7D4A4B4C4D4E4F5Ø5152FB968197A398' D
         DC    X'5CF6535455565758595AFDFFFFE3EEEF' E
         DC    X'3Ø313233343536373839FCEA9AEBE9FF' F
IECTRASA EQU   *                   USASCII INPUT TRANSLATE TABLE
*                 Ø 1 2 3 4 5 6 7 8 9 A B C D E F
         DC    X'ØØØ1Ø2Ø3372D2E2FØ8Ø515ØBØCØDØEØF' Ø
         DC    X'1Ø1112133C3D322618193F271C1D1E1F' 1
         DC    X'4Ø4F7F7B5B6C5Ø7D4D5D5C4E6B6Ø4B61' 2
         DC    X'FØF1F2F3F4F5F6F7F8F97A5E4C7E6E6F' 3
         DC    X'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6' 4
         DC    X'D7D8D9E2E3E4E5E6E7E8E94AEØ5A5F6D' 5
         DC    X'79818283848586878889919293949596' 6
         DC    X'979899A2A3A4A5A6A7A8A9CØ6ADØA1Ø7' 7
         DC    X'68DC5142434447485253545756586367' 8
         DC    X'713334CBCCCDDBDDDFFFFC7ØB18ØBF36' 9
         DC    X'4555CEDE6A7B9A9BABAF5FB8B7BC8A8B' A
         DC    X'Ø4Ø6Ø8Ø9ØA656264B41415171ABØB21B' B
         DC    X'1C1E2Ø212324466628292A2B2C3Ø319F' C
         DC    X'8CAC727374FF75767738393A3B3E7841' D
         DC    X'9C599EEDCFAAAØ8EAEFEFBFD8DADEEEF' E
         DC    X'CA8FFFB9B6B5E19D9ØBEB3DAFAEAFFFF' F
         LTORG
         END   DFH$XCOU
```

Hernán Zamudio Enzian
Systems Programmer (Peru)

CICS trace analysis program – part 2

This month we conclude the code for the CICS trace analysis program.

```
tab.Ø = Ø                            /* result table                 */
if (substr(in.1, 2, 22) ¬= 'CICS - AUXILIARY TRACE') then
   CICS_Trace = 'Y'
if (substr(in.1, 2, 14) ¬= 'IPCS PRINT LOG') then
   IPCS_Trace = 'Y'
if (CICS_Trace ¬= 'Y') & (IPCS_Trace ¬= 'Y') then
do
  call message 'Input dataset 'dsnin' does not contain a CICS '||,
               'formatted Trace with ABBREV!'||,
               '                                                  '||,
               'Use either DFHTUvrm or IPCS to produce a CICS',
               'formatted Trace.'
  signal ende
end
else do
   /*--------------------------------------------------------------*/
  /* Take the header of the trace and put it into output dataset   */
   /*--------------------------------------------------------------*/
  call output in.1
  "EXECIO 1 DISKR CITRIN (STEM in.)"
  execio_rc = rc
  abbrev = 'N'
  if (CICS_Trace = 'Y') then
  do
    do i = 1 to 1ØØ while (execio_rc = Ø & abbrev = 'N')
      if (index(in.1, 'ABBREV') > Ø) then abbrev = 'Y'
      call output in.1
      "EXECIO 1 DISKR CITRIN (STEM in.)"
      execio_rc = rc
    end
  end
  else do
    do i = 1 to 1ØØ while (execio_rc = Ø & abbrev = 'N')
      if (index(in.1, 'ABBREV') > Ø) then abbrev = 'Y'
      if (substr(in.1, 2, 1Ø) = 'CICS DUMP:') then Header = 'Y'
      if (Header = 'Y') then
        call output in.1
      "EXECIO 1 DISKR CITRIN (STEM in.)"
      execio_rc = rc
    end
  end
  if (abbrev = 'N') then
  do
    call message 'Input dataset 'dsnin' is not a CICS trace ',
                 'formatted with parameter ABBREV '
    signal ende
  end
  sdat = substr(date(e),1,2)'.'substr(date(e),4,2)'.'||,
         substr(date(e),7,2)
  call output ' '; call output ' '
  call output ' CITRLINK 2.1.Ø Statistics'
  call output '   Run Date / Time..: 'sdat' / 'time()
  call output '   Input Dataset....: 'dsnin
  call output ' '; call output ' '
end
 /*----------------------------------------------------------------*/
/* Example of a CICS trace record:                                  */
/* 44625    PG Ø9Ø1 PGPG  ENTRY INITIAL_LINK         CI5ØØLO       */
/* 389Ø7 QR XS Ø7ØA XSRC  EVENT CHECK-COMPLETE   CICSTE.MENU userid*/
```

```
 /*--------------------------------------------------------------*/
"EXECIO 1 DISKR CITRIN (STEM in.)"   /* read next record          */
execio_rc = rc
do while (execio_rc = Ø)
  parse var in.1 tasknr tcbname domain traceid gate point function,
                resource
   /*-------------------------------------------------------------*/
  /* If "tasknr" is numeric then a valid CICS trace line has been  */
  /* found.                                                        */
   /*-------------------------------------------------------------*/
  if (datatype(tasknr) = 'NUM') then do
    if (length(tasknr) > 5) then     /* cut off leading ANSI       */
      tasknr = substr(tasknr, 2)     /* number                     */
     /*-----------------------------------------------------------*/
    /* Task start and transaction Id                               */
    /* Put Task number and transaction Id in table                 */
    /* Related trace entry:                                        */
    /* XS Ø7Ø1 XSRC  ENTRY CHECK_CICS_RESOURCE   CSMI,TRANSATTACH  */
     /*-----------------------------------------------------------*/
    if (gate = 'XSRC') & (point = 'ENTRY') &,
       (function = 'CHECK_CICS_RESOURCE')  then
    do
      if (debug = 'ON') then
      call debug u, 'SAY',proc,
                ,'New Task='tasknr
      i = tab.Ø                      /* result table counter + 1   */
      i = i + 1
      tab.i = tasknr
      tab.Ø = i
      tab.tasknr.Ø      = Ø          /* program counter            */
      tab.tasknr.level  = Ø          /* program hierarchy depth    */
      tab.tasknr.tcb       = 'QR'    /* task startes on QR         */
                                     /* this may change on TS 3.1 !*/
      tab.tasknr.tcbswitch = Ø       /* # of TCB switches          */
      tab.tasknr.Ø.EXEC.Ø = Ø        /* number of EXEC CICS        */
      tab.tasknr.Ø.EXEC.sort.Ø = Ø   /* sorted EXEC CICS Cmds      */
      tab.tasknr.Ø.PGM.Ø = Ø         /* number of programs in sort */
      tab.tasknr.Ø.PGM.sort.Ø = Ø    /* sorted program names       */
      parse var resource tab.tasknr.trnid (komma) x1,
                         tab.tasknr.1.traceln
    end
    else do;
       /*---------------------------------------------------------*/
      /* This task number already found ?                          */
       /*---------------------------------------------------------*/
      do i = 1 to tab.Ø
        if (tab.i = tasknr) then
        do
           /*-----------------------------------------------------*/
          /* Call Analyze only for relevant entries                */
```

```
            /*------------------------------------------------------*/
            if (gate = 'PGPG') | (gate = 'APPG') | (gate = 'PGLU') |,
               (gate = 'EIP ') | (gate = 'PGLE') | (gate = 'PGIS') |,
               (gate = 'D2EX1')| (gate = 'ZIS2') | (gate = 'XSRC') |,
               (gate = 'TMP')  | (gate = 'PGLD') then,
            do
              signal ON ERROR   name error /* activate error handling*/
              signal ON SYNTAX  name error
              call analyze_trace
              signal OFF ERROR             /* deactivate error handl.*/
              signal OFF SYNTAX
              i = tab.Ø                    /* Stop loop now          */
            end
          end
        end
      end
    end
    "EXECIO 1 DISKR CITRIN (STEM in.)"
    execio_rc = rc
  end /* do while (execio_rc = Ø)  */
call_output_result:
  "EXECIO Ø DISKR CITRIN (FINIS)"
  execio_rc = rc
  signal ON ERROR   name error_end         /* activate error handling*/
  signal ON SYNTAX  name error_end
  call output_result                       /* Print out the result   */
  signal OFF ERROR
  signal OFF SYNTAX
  call message  tab.Ø' programm calls found'
disp_erg:
  call display
ende:
  if (debug = 'ON') then
    call debug u, 'ENDE', proc
  exit
   /*------------------------------------------------------------*/
   /* Subroutine: ANALYZE_TRACE                                  */
   /* Interpretation and scan of trace records                   */
   /*------------------------------------------------------------*/
analyze_trace:
      /*---------------------------------------------------------*/
      /* Search LINK ENTRY (the program name will be found only at   */
      /* LINK_EXEC entry for local program calls or in the           */
      /* PGIS INQUIRE_PROGRAM Entry for DPLed programs)              */
      /* Related trace entry:                                        */
      /* AP ØØE1 EIP   ENTRY LINK                                    */
      /*---------------------------------------------------------*/
      if (gate = 'EIP') & (point = 'ENTRY') &,
         (function = 'LINK') & (tab.Ø > Ø)  then
      do
        j = tab.tasknr.Ø
        j = j + 1
        tab.tasknr.Ø     = j
        k = tab.tasknr.level
        k = k + 1
        tab.tasknr.level = k
        tab.tasknr.j.traceln = substr(resource, length(resource)-9)
        tab.tasknr.j.Command = 'LINK'
        tab.tasknr.j.EXEC.Ø = Ø           /* number of EXEC CICS     */
        cmd = 'EXEC CICS LINK'
      end
      /*-------------------------------------------------------------*/
      /* Count all EXEC CICS Commands here                          */
      /* Stack them in table tab.tasknr.Ø.EXEC.sort in sort order.   */
      /*                                                            */
      /* Related trace entry:                                       */
      /* AP ØØE1 EIP   ENTRY exec-cics-command                      */
      /* AP 318Ø D2EX1 ENTRY APPLICATION        REQUEST EXEC SQL ... */
      /*-------------------------------------------------------------*/
      if ((gate = 'EIP') | (gate = 'D2EX1')) &,
         (point = 'ENTRY') then,
      do
          j = tab.tasknr.Ø
          k = tab.tasknr.level
        /*say 'tasknr='tasknr 'j='j 'k='k function*/
          if (k < Ø) then nop
          else do
            l=  tab.tasknr.j.EXEC.Ø         /* number of EXEC CICS   */
            l = l + 1
            tab.tasknr.j.EXEC.Ø = l
            if (function ¬= 'LINK') then
            do
              if (gate = 'D2EX1') then      /* DB2 call              */
              do
                if (function = 'APPLICATION') then /* Extract SQL    */
                do
                  parse var resource r1 r2 r3 r4 resource
                  cmd = r2 r3 r4            /* EXEC SQL ....         */
                  /*-----------------------------------------------*/
                  /* For SQL commands switch from QR to L8 if on QR. */
                  /* When already on L8 then do nothing.             */
                  /*-----------------------------------------------*/
                  if (tab.tasknr.tcb = 'QR') then
                  do                        /* switch from QR to L8  */
                    tab.tasknr.tcb       = 'L8'
                    tab.tasknr.tcbswitch = tab.tasknr.tcbswitch + 1
                  end
                end
                else do
                  tab.tasknr.j.EXEC.l = ''
                  signal analyze_trace_return
                end
              end
              else
              do
                cmd = 'EXEC CICS '||strip(function, 'B')
                /*-----------------------------------------------------*/
                /* Set flags for additional information for specific */
                /* EXEC CICS commands                                 */
                /*-----------------------------------------------------*/
                select
                  when(function = 'DELETE'  |,
                       function = 'ENDBR'   |,
                       function = 'READ'    |,
                       function = 'READNEXT'|,
                       function = 'READPREV'|,
                       function = 'RESETBR' |,
                       function = 'REWRITE' |,
                       function = 'STARTBR' |,
                       function = 'UNLOCK'  |,
                       function = 'WRITE'   ) then,
                       tab.tasknr.j.Command = 'VSAM'
                  when(function = 'LOAD'    |,
                       function = 'RELEASE' ) then,
                       tab.tasknr.j.Command = 'LOAD'
                  otherwise tab.tasknr.j.Command = ''
                end
              end
              tab.tasknr.j.EXEC.l = substr(mark, 1, (k)*2)||'| '||,
                                     cmd
              if (r3 = 'SQL') then
              do
                tab.tasknr.j.EXEC.l.traceln = resource
                r3 = ''
              end
              else do
                x = ThreadSafe(cmd)
                /*-----------------------------------------------------*/
                /* mark command and program as non-threadsafe        */
                /*-----------------------------------------------------*/
                if (x = 'No') then
                do
                  tab.tasknr.j.EXEC.l = tab.tasknr.j.EXEC.l||'(!)'
                  /*---------------------------------------------------*/
                  /* check the right program level to mark the right */
                  /* program as non-threadsafe.                       */
                  /*---------------------------------------------------*/
                  if (tab.tasknr.level ¬= tab.tasknr.j.pgm.level) then
                  do
                    do m = j to Ø by -1 until,
                      (tab.tasknr.level = tab.tasknr.m.pgm.level)
                    end
                    if (substr(tab.tasknr.m.pgm, ,
                                 length(tab.tasknr.m.pgm)) ¬= ')') then
                    do
                      call sort_PGM tab.tasknr.m.pgm '(*)'
                      tab.tasknr.m.pgm     = tab.tasknr.m.pgm||'(*)'
                    end
                  end
                  else do
                    if (substr(tab.tasknr.j.pgm, ,
                                 length(tab.tasknr.j.pgm)) ¬= ')') then
                    do
                      call sort_PGM tab.tasknr.j.pgm '(*)'
                      tab.tasknr.j.pgm     = tab.tasknr.j.pgm||'(*)'
                    end
                  end
                  /*--------------------------------------------------*/
                  /* For non-threadsafe commands switch back from     */
                  /* L8 to QR when on L8.                             */
                  /*--------------------------------------------------*/
                  if (tab.tasknr.tcb = 'L8') then
                  do
                    tab.tasknr.tcb       = 'QR'
                    tab.tasknr.tcbswitch = tab.tasknr.tcbswitch + 1
                  end
                end
                tab.tasknr.j.EXEC.l.traceln = ,
                                    substr(resource, length(resource)-9)
              end
            end
            else
              tab.tasknr.j.EXEC.l = ''
            /*--------------------------------------------------------*/
            /* Build sorted table of distinct EXEC CICS commands      */
            /* and count each of them                                 */
            /*--------------------------------------------------------*/
            call Sort_EXEC
          end
        signal analyze_trace_return
      end
      /*--------------------------------------------------------------*/
      /* Search for additional EXEC CICS commands info e.g. the file */
      /* name for VSAM commands.                                      */
      /*--------------------------------------------------------------*/
      /* Related trace entry for VSAM request:                        */
      /* AP EAØØ TMP   ENTRY LOCATE                AFCT,file....      */
      /*--------------------------------------------------------------*/
      if (gate = 'TMP') & (point = 'ENTRY') & (function = 'LOCATE') then
      do
```

```
  j = tab.tasknr.Ø
  if (tab.tasknr.j.Command = 'VSAM') then
  do
    tab.tasknr.j.Command = ''
    l=  tab.tasknr.j.EXEC.Ø           /* number of EXEC CICS   */
    parse var resource table (komma) file resource
    tab.tasknr.j.EXEC.l = tab.tasknr.j.EXEC.l file
  end
end
/*-------------------------------------------------------------*/
/* Related trace entry for LOAD request:                        */
/* PG Ø6Ø1 PGLD  ENTRY LOAD_EXEC             loadmod,.....      */
/* PG Ø6Ø1 PGLD  ENTRY RELEASE_EXEC          loadmod            */
/*-------------------------------------------------------------*/
if (gate = 'PGLD') & (point = 'ENTRY') &,
   ((function = 'LOAD_EXEC') | (function = 'RELEASE_EXEC')) then,
do
  j = tab.tasknr.Ø
  if (tab.tasknr.j.Command = 'LOAD') then
  do
    tab.tasknr.j.Command = ''
    l=  tab.tasknr.j.EXEC.Ø           /* number of EXEC CICS   */
    if (function = 'LOAD_EXEC') then
      parse var resource loadmod (komma) resource
    else
      parse var resource loadmod resource
    tab.tasknr.j.EXEC.l = tab.tasknr.j.EXEC.l strip(loadmod)
  end
end
/*-------------------------------------------------------------*/
/* Search end of program link                                   */
/* Attention: 'EIP   ENTRY RETURN' will also be found when a    */
/* module has been called via 'PGLU  ENTRY LINK_URM'.           */
/* This must not be registerd as a "real" RETURN.               */
/* Related trace entry:                                         */
/* AP ØØE1 EIP   EXIT  LINK                                     */
/* PG ØAØ1 PGLU  ENTRY LINK_URM              HUKDYP,29EDD       */
/*-------------------------------------------------------------*/
if ((gate = 'EIP') | (gate = 'PGLU')) &,
   (point = 'EXIT') &,
   ((function = 'LINK') | (function = 'LINK_URM/OK')) &,
   (tab.Ø > Ø)  then
do
  j = tab.tasknr.Ø
  if (debug = 'ON') then
  call debug u, 'SAY',proc,
           ,'gate='gate 'point='point 'function='function,
            'Level='tab.tasknr.level,
            'Program='tab.tasknr.j.pgm
  if (tab.tasknr.Ø.urm = 'Yes') then
```

```
    tab.tasknr.Ø.urm = 'No'
  else
    tab.tasknr.level = tab.tasknr.level - 1
  if (debug = 'ON') then
    call debug u, 'SAY',proc, 'Level='tab.tasknr.level
  signal analyze_trace_return
end
/*--------------------------------------------------------------*/
/* Search LINK_EXEC with program name                           */
/* Related trace entry:                                         */
/* PG 11Ø1 PGLE  ENTRY LINK_EXEC            pgm____,aaaaaaaa  */
/*--------------------------------------------------------------*/
if (gate = 'PGLE') & (point = 'ENTRY') &,
   (function = 'LINK_EXEC') & (tab.Ø > Ø)  then
do
  j = tab.tasknr.Ø
  k = tab.tasknr.level
  parse var resource resource (komma) rest
  tab.tasknr.j.pgm = strip(resource, 'B')
  call sort_PGM tab.tasknr.j.pgm      /* program name in table */
  tab.tasknr.j.pgm.level = k
  tab.tasknr.j.Command = ''  /* Delete LINK for local program  */
  signal analyze_trace_return
end
/* Search for DPL PROGRAM LINK                                  */
/* Related trace entry:                                         */
/* PG Ø5ØØ PGIS  ENTRY INQUIRE_PROGRAM       pgm_____          */
if (gate = 'PGIS') & (point = 'ENTRY') &,
   (function = 'INQUIRE_PROGRAM') & (tab.Ø > Ø)  then
do
  j = tab.tasknr.Ø
  if (tab.tasknr.j.Command = 'LINK') then
  do
    k = tab.tasknr.level
    parse var resource resource rest
    tab.tasknr.j.pgm = strip(resource, 'L')
    tab.tasknr.j.pgm.level = k
  end
  signal analyze_trace_return
end
/* Search CICS APPLID for DPLed program and check if TCB switch*/
/* Related trace entry:                                         */
/* AP DD21 ZIS2  EVENT IRC                  SWITCH FIRST/SUBSE*/
if (traceid = 'DD21') & (gate = 'ZIS2') & (point = 'EVENT') &,
   (function = 'IRC') & (tab.Ø > Ø)  then
do
  /* Function shipping commands switch back from                */
  /* L8 to QR when on L8.                                       */
  /* So mark even a threasafe command as non-threadsafe if this*/
  /* is a function shipping request.                            */
```

```
      if (tab.tasknr.tcb = 'L8') then
      do
        tab.tasknr.tcb       = 'QR'
        tab.tasknr.tcbswitch = tab.tasknr.tcbswitch + 1
        tab.tasknr.j.EXEC.l = tab.tasknr.j.EXEC.l||'(!)'
        /* check the right program level to mark the right          */
        /* program as non-threadsafe because the program contains  */
        /* a non-threadsafe EXEC CICS command.                      */
        if (tab.tasknr.level ¬= tab.tasknr.j.pgm.level) then
        do
          do m = j to Ø by -1 until,
            (tab.tasknr.level = tab.tasknr.m.pgm.level)
          end
          if (substr(tab.tasknr.m.pgm, ,
                       length(tab.tasknr.m.pgm)) ¬= ')') then
          do
            call sort_PGM tab.tasknr.m.pgm '(*)'
            tab.tasknr.m.pgm     = tab.tasknr.m.pgm||'(*)'
          end
        end
        else do
          if (substr(tab.tasknr.j.pgm, ,
                       length(tab.tasknr.j.pgm)) ¬= ')') then
          do
            call sort_PGM tab.tasknr.j.pgm '(*)'
            tab.tasknr.j.pgm     = tab.tasknr.j.pgm||'(*)'
          end
        end
      end
      j = tab.tasknr.Ø
      if (tab.tasknr.j.Command = 'LINK') then
      do
        tab.tasknr.j.Command = ''
        k = tab.tasknr.level
        parse var resource rest1 (kl_auf) resource (kl_zu) rest
      /*say rest1 resource rest*/
        if (k <= Ø) then nop
        else do
          call sort_PGM tab.tasknr.j.pgm  /* program name in table */
          tab.tasknr.j.pgm = tab.tasknr.j.pgm||' in 'resource
        end
      end
      signal analyze_trace_return
    end
    /* Search for USERID                                            */
    /* Related trace entry:                                         */
    /* XS Ø7ØA XSRC  EVENT CHECK-COMPLETE        userid.CSMI userid*/
    if (gate = 'XSRC') & (point = 'EVENT') &,
       (function = 'CHECK-COMPLETE')       &,
       (tab.Ø > Ø)                         then
      do
        parse var resource x1 tab.tasknr.userid x2
        signal analyze_trace_return
      end
      /* Search for Initial Program                                  */
      /* Related trace entry:                                        */
      /* PG Ø9Ø1 PGPG  ENTRY INITIAL_LINK          pgm.....          */
      if ((gate = 'PGPG') | (gate = 'APPG')) &,
         ((point = 'ENTRY') & (function = 'INITIAL_LINK'))   then,
      do
        j = tab.tasknr.Ø
        if (tab.tasknr.Ø.urm = 'Yes') then nop
        else do
          j = j + 1
          tab.tasknr.Ø     = j
        end
        tab.tasknr.j.EXEC.Ø = Ø           /* number of EXEC CICS     */
        tab.tasknr.level = Ø
        parse var resource tab.tasknr.j.pgm tracel
        call sort_PGM tab.tasknr.j.pgm    /* program name in table   */
        tab.tasknr.j.pgm.level = Ø
        tab.tasknr.j.traceln = strip(tracel)
        signal analyze_trace_return
      end
      /* Check LINK_URM (eg. for Autoinstall for programs)           */
      /* Related trace entry:                                        */
      /* PG ØAØ1 PGLU  ENTRY LINK_URM              urm_name          */
      if (gate = 'PGLU') & (point = 'ENTRY') &,
         (function = 'LINK_URM') & (tab.Ø > Ø)  then
      do
        tab.tasknr.Ø.urm = 'Yes'
        j = tab.tasknr.Ø
        j = j + 1
        tab.tasknr.Ø     = j
        tab.tasknr.j.EXEC.Ø = Ø           /* number of EXEC CICS     */
        k = tab.tasknr.level
        parse var resource resource (komma) r1 r2 r3 tracel
        tab.tasknr.j.pgm = strip(resource, 'L')
        call sort_PGM tab.tasknr.j.pgm    /* program name in table   */
        tab.tasknr.j.pgm.level = k
        tab.tasknr.j.traceln = strip(tracel)
        signal analyze_trace_return
      end
analyze_trace_return:
return
   /* Subroutine OUTPUT                                              */
   /* Environment dependent output of result.                        */
output:
  arg daten
  parse arg daten
  if (env = 'TSO') | (env = 'NTSO') then
    say substr(daten, 2)
    out.o = daten
    o = o + 1                          /* increase line counter      */
return
   /* Subroutine MESSAGE                                             */
   /* Environment dependent output of a message.                     */
MESSAGE:
  arg daten
  parse arg daten
  if (env = 'TSO') | (env = 'NTSO') then
    say daten
  else do
    zedlmsg = daten
    address ISPEXEC "SETMSG  MSG(ISRZØØ1)"
  end
return
   /* Subroutine ALLOK                                               */
   /* Allocate an Input dataset.                                     */
   /* Environment dependent allocation of a input dataset.           */
allok:
  parse arg para dsn
  rc = Ø
  if (para = 'INPUT') then
  do
    dsnin = dsn_build(dsn)
    x = outtrap('outline.')
    x = msg('OFF')
    sysdsn_msg = sysdsn("'"dsnin"'")
    /* Check whether Input DSN exists.                           */
    if (sysdsn_msg = 'OK') then
    do
      address TSO "FREE",
             "DDNAME(CITRIN)"
      address TSO "ALLOCATE",
             "DDNAME(CITRIN) DA('"dsnin"') SHR"
    end
    else do
      rc = 8
      outline.1 = sysdsn_msg; outline.2 = ''
      signal allok_ende
    end
    x = outtrap('OFF')
    x = msg('ON')
  end
  else do
    if (env ¬= 'TSO') & (env ¬= 'NTSO') then
    do
      dsntemp = userid()'.CITRLINK.TEMP'
      if (sysdsn("'"dsntemp"'") ¬= 'OK') then
      do
        address tso "ALLOCATE",
               "DDNAME(CITROUT) DA('"dsntemp"') NEW      CATALOG",
               "DSORG(PS)   RECFM(F,B,A)   LRECL(8Ø)",
               "CYLINDERS SPACE(1,1)  REUSE"
      end
      else do
        address tso "ALLOCATE",
               "DDNAME(CITROUT) DA('"dsntemp"') SHR"
      end
    end
  end
allok_ende:
return
   /* Subroutine DISPLAY                                              */
   /* Environment dependent display of the result dataset.            */
display:
  if (env ¬= 'TSO') & (env ¬= 'NTSO') then
  do
    if (o > 1) then
    do
      /* output Stem and free datasets                                */
      address TSO "EXECIO * DISKW CITROUT (STEM OUT. finis)"
      address TSO "FREE DDNAME(CITROUT)"
      /* Show result dataset                                          */
      address ISPEXEC
      zedsmsg = ''
      zedlmsg = 'CICS Trace analysis completed.'
      "SETMSG  MSG(ISRZØØ1)"
      "CONTROL DISPLAY SAVE"
      "CONTROL ERRORS RETURN"
      "BROWSE  DATASET('"dsntemp"')"
      "CONTROL DISPLAY RESTORE"
      "CONTROL ERRORS CANCEL"
    end /* if (o > 1) then */
    else
      address TSO "FREE DDNAME(CITROUT)"
  end /* if (env ¬= 'TSO') & (env ¬= 'NTSO') then */
  else
      address TSO "EXECIO * DISKW CITROUT (STEM OUT. finis)"
  address TSO "FREE DDNAME(CITRIN)"
return
  /* Subroutine DSN_BUILD                                             */
  /* This subroutine returns a valid DSN without quotation marks.     */
  /* The setting of TSO PROFILE NO/PREFIX will be checked and used    */
  /* to generate the DSN which follows the ISPF standard convention. */
  /* Thus the userid will not be set before the DSN if specified in  */
  /* the   TSO PROFILE.                                               */
DSN_BUILD:
  arg para
  prefix = sysvar(syspref)
  if (substr(para,1,1) =  "'") then
    dsn = strip(para, B, "'")
  else do
    if (prefix ¬= '') then  dsn = prefix'.'para
    else dsn = para
  end
return_DSN_BUILD:
return dsn
/* Subroutine error                                                  */
/* Unpredictable errors will cause an invocation of this routine to  */
/* help locating the error.                                          */
error:
  in_error_jump = 'error'
error_end:
  if (in_error_jump = '') then
    in_error_jump = 'error_end'
  rc_error = rc
                                     /* No error at EXECIO or EOF    */
  if (rc_error ¬= 1) & (rc_error ¬= 2) then
  do
  say '------------------------------------'
  say 'Here is the error handle from EXEC: 'proc
  say '------------------------------------'; say
  say 'REXX error       : 'rc 'in line' sigl
  say 'REXX error text  : 'errortext(rc)
  say '                   'substr(sourceline(value(sigl-5)),1, 59)
  say '                   'substr(sourceline(value(sigl-4)),1, 59)
  say '                   'substr(sourceline(value(sigl-3)),1, 59)
  say '                   'substr(sourceline(value(sigl-2)),1, 59)
  say '                   'substr(sourceline(value(sigl-1)),1, 59)
  say 'Error instruction: 'substr(sourceline(sigl),1, 59)
  say '                   'substr(sourceline(value(sigl+1)),1, 59)
  say '                   'substr(sourceline(value(sigl+2)),1, 59)
  say '                   'substr(sourceline(value(sigl+3)),1, 59)
  say '                   'substr(sourceline(value(sigl+4)),1, 59)
  say '                   'substr(sourceline(value(sigl+5)),1, 59)
  say
  parse var resource (ist) tracel   /* parse for trace number  */
  say 'Current trace number in input dataset 'dsnin' is '
  say
  say '        'ist||strip(tracel)
  say
  say
  say 'Content of line 'ist||strip(tracel)||' is:'
  say ''in.1''
/*signal OFF ERROR
  signal OFF SYNTAX */
  end
  if (in_error_jump = 'error_end') then
  do
    say ' '
    say ' '
    say ' '
    say 'An error has occurred during print processing.'
    say ' '
    say 'Despite this the result so far will be shown!'
    say ' '
    signal disp_erg                    /* Show result so far          */
  end
  else
    signal call_output_result
  in_error_jump = ''
return
  /* Subroutine CHECK_PARMS                                           */
  /* Check input parameters and set their variables.                  */
CHECK_PARMS:
  arg parm1 parm2
  rc = Ø
  select
    when (parm1 = '') then
    do
      call message 'You must give the dataset name of the CICS ABBREV',
                   'Trace as a parameter.'
      rc = 12
    end
    when (parm1 = '?') then
    do
      s  =  1                          /* start position              */
      ll = 72                          /* length of comment line      */
      n  = length(doc) / ll            /* number of lines             */
      do i = 1 to n
        say substr(doc, s, ll)
        s = s + ll
      end
      rc = 4
    end
    otherwise nop                      /* no further checking here    */
  end
  if rc > Ø then signal return_CHECK_PARMS
  select
    when (parm2 = '') then nop
    when (parm2 = '-D') then
      detail = 'Y'                     /* Detail statistics           */
    otherwise do
      call message 'Wrong parameter value 'parm2'. Right values are:',
                   '                                              ',
                   '-D  - Detail statistics'
      rc = 8
    end
  end
return_CHECK_PARMS:
return rc
  /* Subroutine    Sort_EXEC                                         */
  /* Brings an EXEC CICS into a sorted table. Each command will      */
  /* have one table entry and a counter which holds the command's    */
  /* usage number.                                                   */
Sort_EXEC:
trace off
            if (tab.tasknr.Ø.EXEC.sort.Ø = Ø) then /* 1st entry ?    */
            do
              tab.tasknr.Ø.EXEC.sort.Ø    = 1
              tab.tasknr.Ø.EXEC.sort.1    = cmd
              tab.tasknr.Ø.EXEC.sort.1.Ø = 1        /* cmd 1 times   */
              signal return_Sort_EXEC
            end
            else
            do
              do i = 1 to tab.tasknr.Ø.EXEC.sort.Ø
                if (cmd = tab.tasknr.Ø.EXEC.sort.i) then
                do
                  tab.tasknr.Ø.EXEC.sort.i.Ø = ,
                  tab.tasknr.Ø.EXEC.sort.i.Ø + 1
                  signal return_Sort_EXEC
                end
                else do
                  if (cmd < tab.tasknr.Ø.EXEC.sort.i) then
                  do
                    /* shift rest of table one entry                 */
                    k = tab.tasknr.Ø.EXEC.sort.Ø + 1 /* new last     */
                    l = k - 1
                    do j = i to tab.tasknr.Ø.EXEC.sort.Ø
                      tab.tasknr.Ø.EXEC.sort.k =,
                                         tab.tasknr.Ø.EXEC.sort.l
                      tab.tasknr.Ø.EXEC.sort.k.Ø =,
                                         tab.tasknr.Ø.EXEC.sort.l.Ø
                      k = k - 1
                      l = l - 1
                    end
                    /* insert new entry                              */
                    tab.tasknr.Ø.EXEC.sort.i = cmd
                    tab.tasknr.Ø.EXEC.sort.i.Ø = 1
                    tab.tasknr.Ø.EXEC.sort.Ø = ,
                                 tab.tasknr.Ø.EXEC.sort.Ø + 1
                    signal return_Sort_EXEC
                  end
                end
              end
              /* new last table entry                                */
              i                          = tab.tasknr.Ø.EXEC.sort.Ø + 1
              tab.tasknr.Ø.EXEC.sort.Ø = i
              tab.tasknr.Ø.EXEC.sort.i = cmd
              tab.tasknr.Ø.EXEC.sort.i.Ø = 1
            end
return_Sort_EXEC:
return
  /* Subroutine    Sort_PGM                                        */
  /* Brings a program name into a sorted table. Each pgm name will   */
  /* have one table entry and a counter which holds the program name */
  /* name's usage number.                                            */
Sort_PGM:
  arg program pgm_mark
          if (pgm_mark = '') then
          do
            if (tab.tasknr.Ø.PGM.sort.Ø = Ø) then /* 1st entry ?      */
            do
              tab.tasknr.Ø.PGM.sort.Ø      = 1
              tab.tasknr.Ø.PGM.sort.1      = program
              tab.tasknr.Ø.PGM.sort.1.Ø = 1          /* pgm 1 times    */
              tab.tasknr.Ø.PGM.sort.1.mark = ''
              signal return_Sort_PGM
            end
            else
            do
              do x = 1 to tab.tasknr.Ø.PGM.sort.Ø
                if (program = tab.tasknr.Ø.PGM.sort.x) then
                do
                  tab.tasknr.Ø.PGM.sort.x.Ø = ,
                  tab.tasknr.Ø.PGM.sort.x.Ø + 1
                  signal return_Sort_PGM
                end
                else do
                  if (program < tab.tasknr.Ø.PGM.sort.x) then
                  do
                    /* shift rest of table one entry                  */
                    y = tab.tasknr.Ø.PGM.sort.Ø + 1 /* new last       */
                    z = y - 1
                    do w = x to tab.tasknr.Ø.PGM.sort.Ø
                      tab.tasknr.Ø.PGM.sort.y =,
                                         tab.tasknr.Ø.PGM.sort.z
                      tab.tasknr.Ø.PGM.sort.y.Ø =,
                                         tab.tasknr.Ø.PGM.sort.z.Ø
                      tab.tasknr.Ø.PGM.sort.y.mark =,
                                         tab.tasknr.Ø.PGM.sort.z.mark
                      y = y - 1
                      z = z - 1
                    end
                    /* insert new entry                               */
                    tab.tasknr.Ø.PGM.sort.x = program
                    tab.tasknr.Ø.PGM.sort.x.Ø = 1
```

```
                      tab.tasknr.Ø.PGM.sort.x.mark = ''
                      tab.tasknr.Ø.PGM.sort.Ø = ,
                                   tab.tasknr.Ø.PGM.sort.Ø + 1
                      signal return_Sort_PGM
                    end
                  end
                end
                /* new last table entry                                */
                x                        = tab.tasknr.Ø.PGM.sort.Ø + 1
                tab.tasknr.Ø.PGM.sort.Ø = x
                tab.tasknr.Ø.PGM.sort.x = program
                tab.tasknr.Ø.PGM.sort.x.mark = ''
                tab.tasknr.Ø.PGM.sort.x.Ø = 1
              end
            end /* if (pgm_mark = '') then */
            else do
              do x = 1 to tab.tasknr.Ø.PGM.sort.Ø until,
                (program = tab.tasknr.Ø.PGM.sort.x)
              end
              tab.tasknr.Ø.PGM.sort.x.mark = pgm_mark
            end
return_Sort_PGM:
return
  /* Subroutine    Sort_PGM_Used                                      */
  /* This subroutine generates a second sorted table for programs.   */
  /* They will get there only during output operation and are in     */
  /* decending order by use count and ascending by program name.     */
Sort_PGM_Used:
          tab.PGM.sort.Ø = Ø
          do u = 1 to tab.tasknr.Ø.PGM.sort.Ø
            /* Build program name and count ccccc.                    */
            program = tab.tasknr.Ø.PGM.sort.u
            if (tab.tasknr.Ø.PGM.sort.u.mark ¬= '') then
              program = program||tab.tasknr.Ø.PGM.sort.u.mark
            p  = 5 - length(tab.tasknr.Ø.PGM.sort.u.Ø) + 1
            count = overlay(tab.tasknr.Ø.PGM.sort.u.Ø, '     ', p)
            if (tab.PGM.sort.Ø = Ø) then /* 1st entry ?               */
            do
              tab.PGM.sort.Ø             = 1
              tab.PGM.sort.1.cnt         = count
              tab.PGM.sort.1.pgm         = program
            end
            else
            do
              do x = 1 to tab.PGM.sort.Ø
                if (count > tab.PGM.sort.x.cnt) then
                do
                    /* shift rest of table one entry                  */
                    y = tab.PGM.sort.Ø + 1 /* new last                */
                    z = y - 1
```

```
    do w = x to tab.PGM.sort.0
      tab.PGM.sort.y.cnt     = tab.PGM.sort.z.cnt
      tab.PGM.sort.y.pgm = tab.PGM.sort.z.pgm
      y = y - 1
      z = z - 1
    end
    /* insert new entry                              */
    tab.PGM.sort.x.cnt     = count
    tab.PGM.sort.x.pgm = program
    tab.PGM.sort.0 = tab.PGM.sort.0 + 1
    x = tab.PGM.sort.0     /* Stop loop now */
end
else do
  if (count = tab.PGM.sort.x.cnt) then
  do
    do v = x to tab.PGM.sort.0 while,
       count = tab.PGM.sort.v.cnt
    end
    if (v >= tab.PGM.sort.0) then
    do
      /* new last table entry                        */
      x              = tab.PGM.sort.0 + 1
      tab.PGM.sort.0 = x
      tab.PGM.sort.x.cnt     = count
      tab.PGM.sort.x.pgm = program
    end
    else do
      /* shift rest of table one entry               */
      y = tab.PGM.sort.0 + 1 /* new last             */
      z = y - 1
      do w = v to tab.PGM.sort.0
        tab.PGM.sort.y.cnt     = tab.PGM.sort.z.cnt
        tab.PGM.sort.y.pgm = tab.PGM.sort.z.pgm
        y = y - 1
        z = z - 1
      end
      /* insert new entry                            */
      tab.PGM.sort.v.cnt     = count
      tab.PGM.sort.v.pgm = program
      tab.PGM.sort.0 = tab.PGM.sort.0 + 1
      x = tab.PGM.sort.0     /* Stop loop now */
    end
  end
  else do
    if (x = tab.PGM.sort.0) then
    do
       /* new last table entry                        */
      x              = tab.PGM.sort.0 + 1
      tab.PGM.sort.0 = x
      tab.PGM.sort.x.cnt     = count
```

```
                      tab.PGM.sort.x.pgm = program
                    end
                  end
                end
              end
            end
          end /* do u = 1 to tab.tasknr.Ø.PGM.sort.Ø */
return_Sort_PGM_Used:
return
  /* Subroutine output_result                                        */
  /* Print out all sampled tables.                                   */
output_result:
  /* Output the result                                               */
  /*  - tab.        contains all found task numbers                  */
  /*  - tab.tasknr. contains per task number all detail information  */
  call output ' '
  call output ' '
  call output ' The following ('tab.Ø') tasks'||,
              ' have been found:'
  do i = 1 to tab.Ø
    tasknr = tab.i
    call output '   '||tab.i||'  Tranid='strip(tab.tasknr.trnid)||,
                ', Called programs: 'tab.tasknr.Ø
  end
  call output ' '
  call output ' All programs marked with (*) contain non-threadsafe'||,
              ' CICS commands|'
  call output ' '
  if (detail = 'Y') then
  do
    call output ' All EXEC CICS Commands marked with (|) are'||,
                ' non-threadsafe CICS Commands|'
    call output ' '
  end
  do i = 1 to tab.Ø
    tasknr = tab.i
    call output ' '
    call output ' '
    call output '1Taskid='tasknr', Tranid='strip(tab.tasknr.trnid)||,
                ', Userid='tab.tasknr.userid
    call output ' '
    do j = 1 to tab.tasknr.Ø
      k = tab.tasknr.j.pgm.level
    /*say 'k='k 'j='j
      say tab.tasknr.j.pgm*/
      if (k = Ø) then
        out = ' '||tab.tasknr.j.pgm
      else
        out = ' '||substr(mark, 1, (k)*2)||tab.tasknr.j.pgm
      out = overlay(strip(tab.tasknr.j.traceln,'B'), out, 73)
      call output out
      /* Detail statistics with all EXEC CICS and EXEC SQL           */
      if (detail = 'Y') then
      do
        do l = 1 to tab.tasknr.j.EXEC.Ø /* Print all EXEC CICS Cmds */
          if (tab.tasknr.j.EXEC.l ¬= '') then
          do
            out = ' '||tab.tasknr.j.EXEC.l
            out = overlay(strip(tab.tasknr.j.EXEC.l.traceln,'B'), ,
                                                              out, 73)
            call output out
          end
        end
      end
    end
    if (tab.tasknr.level = Ø) then
    do
      call output ' 'tab.tasknr.1.pgm||'_END'
      call output ' '
      call output ' '
      call output ' '
      CALL OUTPUT ' OVERVIEW OF USED EXEC CICS COMMANDS AND USE COUNTS'
      call output ' '
      if (tab.tasknr.Ø.EXEC.sort.Ø <= Ø) then
      do
      call output '   No CICS Commands have been found for this '||,
                     'transaction.'
      call output '   Possible cause: No trace entries found from '||,
                     'the EIP component.'
      call output '   See CETR - Components.'
      call output ' '
      call output '   So no further analysis can be shown.'
      call output ' '
      call output ' '
      end
      else do
      call output '   CICS Commands                    Count Threadsafe '
      call output '+  ________________________________  _____ ___________ '
      dotline = '. . . . . . . . . . . . . . . . '
      All_Commands   = Ø
      Threadsafe_yes = Ø
      Threadsafe_no  = Ø
      do j = 1 to tab.tasknr.Ø.EXEC.sort.Ø while,
                 (substr(tab.tasknr.Ø.EXEC.sort.j,1,9) = 'EXEC CICS')
        command = tab.tasknr.Ø.EXEC.sort.j
        x =  ThreadSafe(command)
        if (x = 'Yes') then
          Threadsafe_yes = Threadsafe_yes + tab.tasknr.Ø.EXEC.sort.j.Ø
        else
          Threadsafe_no  = Threadsafe_no  + tab.tasknr.Ø.EXEC.sort.j.Ø
        All_Commands     = All_Commands   + tab.tasknr.0.EXEC.sort.j.0
        command = overlay(command, dotline, 1)
        p  = 5 - length(tab.tasknr.0.EXEC.sort.j.0) + 1
        count = overlay(tab.tasknr.0.EXEC.sort.j.0, '     ', p)
        call output '   'command   count  x
      end
      call output '+  _____________________________  _____ __________ '
      p  = 5 - length(All_Commands) + 1
      count = overlay(All_Commands, '     ', p)
      call output '   *Total EXEC CICS*             '   count
      p  = 5 - length(Threadsafe_yes) + 1
      count = overlay(Threadsafe_yes, '     ', p)
      pct = format((Threadsafe_yes / All_Commands * 100), 3, 1) '%'
      call output '   *Total Threadsafe Cmds*       '   count pct
      p  = 5 - length(Threadsafe_no) + 1
      count = overlay(Threadsafe_no, '     ', p)
      pct = ''
      if (Threadsafe_no > 0) then
        pct = format((Threadsafe_no  / All_Commands * 100), 3, 1) '%'
      else
        pct = ''
      call output '   *Total non-Threadsafe Cmds*   '   count pct
      call output ' '
      call output ' '
      call output ' '
      if (j > tab.tasknr.0.EXEC.sort.0) then
      do
        call output ' No EXEC SQL Commands have been used'
      end
      else do
        CALL OUTPUT ' OVERVIEW OF USED EXEC SQL COMMANDS AND USE COUNTS'
        call output ' '
        call output '   SQL Commands                   Count'
        call output '+  _____________________________  _____'
        All_Commands   = 0
        do j = j to tab.tasknr.0.EXEC.sort.0
          command = tab.tasknr.0.EXEC.sort.j
          All_Commands   = All_Commands   + tab.tasknr.0.EXEC.sort.j.0
          command = overlay(command, dotline, 1)
          p  = 5 - length(tab.tasknr.0.EXEC.sort.j.0) + 1
          count = overlay(tab.tasknr.0.EXEC.sort.j.0, '     ', p)
          call output '   'command   count
        end
        call output '+  _____________________________  _____'
        p  = 5 - length(All_Commands) + 1
        count = overlay(All_Commands, '     ', p)
        call output '   *Total EXEC SQL*              '   count
        call output ' '
        call output ' '
        call output ' '
        CALL OUTPUT ' OVERVIEW OF CALCULATED TCB SWITCHES QR - L8'
        call output ' '
        max_TCB_switches = All_Commands * 2
        p  = 5 - length(max_TCB_switches) + 1
        count = overlay(max_TCB_switches, '     ', p)
        call output '   *Total switches in CICS TS 1.3 and below*      ',
                        count
        p  = 5 - length(tab.tasknr.tcbswitch) + 1
        count = overlay(tab.tasknr.tcbswitch, '     ', p)
        call output '   *Total switches in CICS TS 2.2 and above*      ',
                        count
        call output '    (when all programs are defined with '
        call output '     COncurency(Threadsafe).)'
        reduction = max_TCB_switches - tab.tasknr.tcbswitch
        pct = format((reduction / max_TCB_switches * 100), 3, 1) '%'
        call output ' '
        call output '   *This means a possible reduction of*           ',
                        pct
      end
      call output ' '
      call output ' '
      call output ' '
      CALL OUTPUT ' PROGRAM INDEX:'
      CALL OUTPUT '   SORTED BY PROGRAM NAME                 '||,
                  ' SORTED DESCENDING BY PROGRAM USAGE'
      call output ' '
      call output '   Pgm name      Count                  '||,
                  '   Pgm name      Count'
      call output '+  ___________   _____                  '||,
                  '   ___________   _____'
      PGM_out = '           '
      All_PGM_Calls  = 0
      call Sort_PGM_Used  /* Sort pgms descending after use count    */
                          /* into table tab.pgm.sort                 */
      do j = 1 to tab.tasknr.0.PGM.sort.0
        program = tab.tasknr.0.PGM.sort.j
        if (tab.tasknr.0.PGM.sort.j.mark ¬= '') then
          program = program||tab.tasknr.0.PGM.sort.j.mark
        program = overlay(program, PGM_out, 1)
        All_PGM_Calls = All_PGM_Calls + tab.tasknr.0.PGM.sort.j.0
        p  = 5 - length(tab.tasknr.0.PGM.sort.j.0) + 1
        count = overlay(tab.tasknr.0.PGM.sort.j.0, '     ', p)
        program2 = overlay(tab.pgm.sort.j.pgm, PGM_out, 1)
        call output '   'program ' '  count '                     ' ||,
                         program2 ' '  tab.pgm.sort.j.cnt
      end
      call output '+  ___________   _____                  '||,
                  '   ___________   _____'
      p  = 5 - length(All_PGM_Calls) + 1
      count = overlay(All_PGM_Calls, '     ', p)
```

```
      call output '   *Total*      '  count
      call output ' '
      call output ' '
      end /* if (tab.tasknr.Ø.EXEC.sort.Ø <= Ø) then ... else do */
    end /* if (tab.tasknr.level = Ø) then */
    else
      call output ' ==> Attention: End of task not found in',
                  'CICS trace. Data may be incomplete.'
  end
return
  /* Subroutine REXXENV                                             */
  /* Determine the runtime environment.                             */
REXXENV:
ENVTYPE  = "TSO NTSO ISPF NISPF MACRO NMACRO"
RCODE    = Ø
MACROARG = ""
PROCARG  = ""
parse upper arg NESTED,CALLER,ENVIRON,PROCARG
ISPF = sysvar(sysispf)
if ISPF = 'ACTIVE' then do                  /* Falls Macro RC = Ø und */
  address ispexec "control errors return"    /* Parms in Variable     */
  address isredit "macro (MACROARG)"       /* off here then als Macro */
  MACRO = rc
  address isredit "(MACVAR) = MACRO_LEVEL"     /* Procedure die von  */
  if MACVAR > Ø then MACRO = Ø               /* Edit-MACROs invokes */
                                               /* because MACRO = 2Ø*/
                                                /*  MACVAR > Ø      */
  address ispexec "control errors cancel"
end
 /* Determines the Umgebungs-Typs                                    */
select
  when ISPF ¬= 'ACTIVE' & NESTED = 'YES' then RUNTYPE = 'NTSO'
when ISPF ¬= 'ACTIVE' & NESTED = 'NO'  then RUNTYPE =             'TSO'
when ISPF =  'ACTIVE' & NESTED = 'NO'  & MACRO > Ø then RUNTYPE = 'ISPF'
when ISPF =  'ACTIVE' & NESTED = 'NO' & MACRO = Ø then RUNTYPE = 'MACRO'
when ISPF = 'ACTIVE' & NESTED = 'YES' & MACRO > Ø then RUNTYPE = 'NISPF'
when ISPF = 'ACTIVE' & NESTED = 'YES' & MACRO = Ø then RUNTYPE =
'NMACRO'
end
 /* Only directly invoked Macros have Command-Line-Parameter       */
if RUNTYPE = 'MACRO' then ARGS = MACROARG
else ARGS = PROCARG
do I = 1 to words(ENVIRON)             /* correct Exclude-Environ   */
  if wordpos(word(ENVIRON,I),ENVTYPE,1) = Ø then do
    RCODE = 16
    leave
  end
end
I = 1
do while RCODE = Ø & I <= words(ENVIRON)
```

```
  if RUNTYPE = word(ENVIRON,I) then do
    if left(strip(ARGS,B,' '),1) ¬= '?' then do
      ARGS = '?'
      call CONCAT
    end
    RCODE = 2Ø
  end
  I = I + 1
end
END_REXXENV:
 /*  call DEBUG 'REXX', 'ENDE', proc                    D E B U G P    */
return RCODE RUNTYPE ARGS
  /* Subroutine ThreadSafe                                              */
  /* Check whether input parameter is a Threadsafe command or not      */
  /* Some threadsafe commands will switch from L8 to QR if they use    */
  /* function shipping (see Redbook "Threadsafe considerations for     */
  /* CICS" - SG24-6951), eg LINK, READQ-TS etc.                        */
  /* The list of threadsave commands in this routine is taken from     */
  /* "CICS TS 2.3 Application Programming Reference, Appendix L"       */
  /* "CICS TS 2.3 System Programming Reference, Appendix D"            */
  /* For performance reasons the table is sorted in that way that the*/
  /* most used commands are at the top of the table (performance).     */
  /* Note: The format of the commands is different for some commands */
  /* as printed in the CICS books. The format coded below is the       */
  /* format that CICS produces in the trace records.                   */
ThreadSafe:
  arg para
  para = strip(para, 'B')
  select
    when (para = 'EXEC CICS LINK')                 then threads = 'Yes'
    when (para = 'EXEC CICS RETURN')               then threads = 'Yes'
    when (para = 'EXEC CICS ASSIGN')               then threads = 'Yes'
    when (para = 'EXEC CICS GETMAIN')              then threads = 'Yes'
    when (para = 'EXEC CICS FREEMAIN')             then threads = 'Yes'
    when (para = 'EXEC CICS ABEND')                then threads = 'Yes'
    when (para = 'EXEC CICS ADDRESS')              then threads = 'Yes'
    when (para = 'EXEC CICS ASKTIME')              then threads = 'Yes'
    when (para = 'EXEC CICS CHANGE-TASK')          then threads = 'Yes'
    when (para = 'EXEC CICS DELETEQ-TS')           then threads = 'Yes'
    when (para = 'EXEC CICS DEQ')                  then threads = 'Yes'
    when (para = 'EXEC CICS DOCUMENT-CREATE')      then threads = 'Yes'
    when (para = 'EXEC CICS DOCUMENT-INSERT')      then threads = 'Yes'
    when (para = 'EXEC CICS DOCUMENT-RETRIEVE')    then threads = 'Yes'
    when (para = 'EXEC CICS DOCUMENT-SET')         then threads = 'Yes'
    when (para = 'EXEC CICS ENQ')                  then threads = 'Yes'
    when (para = 'EXEC CICS ENTER-TRACENUM')       then threads = 'Yes'
    when (para = 'EXEC CICS FORMATTIME')           then threads = 'Yes'
    when (para = 'EXEC CICS HANDLE-ABEND')         then threads = 'Yes'
    when (para = 'EXEC CICS HANDLE-AID')           then threads = 'Yes'
    when (para = 'EXEC CICS HANDLE-CONDITION')     then threads = 'Yes'
    when (para = 'EXEC CICS IGNORE-CONDITION')      then threads = 'Yes'
    when (para = 'EXEC CICS LOAD')                  then threads = 'Yes'
    when (para = 'EXEC CICS MONITOR-POINT')         then threads = 'Yes'
    when (para = 'EXEC CICS POP-HANDLE')            then threads = 'Yes'
    when (para = 'EXEC CICS PUSH-HANDLE')           then threads = 'Yes'
    when (para = 'EXEC CICS READQ-TS')              then threads = 'Yes'
    when (para = 'EXEC CICS RELEASE')               then threads = 'Yes'
    when (para = 'EXEC CICS SUSPEND')               then threads = 'Yes'
    when (para = 'EXEC CICS WAIT-EXTERNAL')         then threads = 'Yes'
    when (para = 'EXEC CICS WRITEQ-TS')             then threads = 'Yes'
    when (para = 'EXEC CICS XCTL')                  then threads = 'Yes'
    /* SPI- Commands                                                  */
    when (para = 'EXEC CICS DISCARD-DB2CONN')       then threads = 'Yes'
    when (para = 'EXEC CICS DISCARD-DB2ENTRY')      then threads = 'Yes'
    when (para = 'EXEC CICS DISCARD-DB2TRAN')       then threads = 'Yes'
    when (para = 'EXEC CICS DISCARD-DOCTEMPLATE')   then threads = 'Yes'
    when (para = 'EXEC CICS INQUIRE-DB2CONN')       then threads = 'Yes'
    when (para = 'EXEC CICS INQUIRE-DB2ENTRY')      then threads = 'Yes'
    when (para = 'EXEC CICS INQUIRE-DB2TRAN')       then threads = 'Yes'
    when (para = 'EXEC CICS INQUIRE-DOCTEMPLATE')   then threads = 'Yes'
    when (para = 'EXEC CICS INQUIRE-EXITPROGRAM')   then threads = 'Yes'
    when (para = 'EXEC CICS INQUIRE-TASK')          then threads = 'Yes'
    when (para = 'EXEC CICS INQUIRE-WORKREQUEST')   then threads = 'Yes'
    when (para = 'EXEC CICS SET-DB2CONN')           then threads = 'Yes'
    when (para = 'EXEC CICS SET-DB2ENTRY')          then threads = 'Yes'
    when (para = 'EXEC CICS SET-DB2TRAN')           then threads = 'Yes'
    when (para = 'EXEC CICS SET-WORKREQUEST')       then threads = 'Yes'
    otherwise threads = 'No'
  end
return threads
```

Hans-Joachim Gerdes
CICS Systems Programmer
HUK-COBURG Versicherungen (Germany)

CICS news

IBM has announced CICS TG V6.0.1 and CICS Universal Client (CICS UC) V6.0.1. Both products now support Linux on POWER and AIX (including V5.3).

In addition, they both now support Red Hat Enterprise Linux (RHEL) 4.

IBM claims that this service release can also deliver significant run-time performance enhancements for request processing. The enhancement is provided in the interface to the Client daemon.

For further information contact:
URL: http://www-306.ibm.com/common/ssi/fcgi-bin/ssialias?subtype=ca&infotype=an&appname=iSource&supplier=897&letternum=ENUS205-147.

* * *

IBM has announced Version 1.2 of its CICS VSAM Transparency (CICS VT) for z/OS. The product enables the migration of data from VSAM files to DB2 tables. It ensures continued access to the data in DB2, without modification to the CICS or batch VSAM application programs and with only minor changes to CICS and batch configurations.

CICS VT is a set of conversion tools and a run-time management control component. The conversion tools include all the utilities required for conversion of VSAM KSDS and RRDS files to DB2 tables, the generation of the DB2 SQL drivers, and the one time migration of data into DB2 from VSAM.

The run-time component intercepts the application program's original and unaltered VSAM calls, processes them in the generated DB2 SQL driver, and returns the results to the calling program.

The new version includes the following features: Dual Mode Facility, which assists with the testing of migrated data and in the development of user exits; enhancements to the mapping component, which now supports the Auto Mapper feature for PL/I and Assembler copybooks; and performance enhancements, which enable the CICS VT-generated SQL driver modules to be re-entrant.

For further information contact:
URL: www-306.ibm.com/common/ssi/fcgi-bin/ssialias?subtype=ca&infotype=an&appname=iSource&supplier=897&letternum=ENUS205-148.

* * *

A new white paper entitled "Making CICS a peer layer in Web services" is available from IBM. It describes how CICS Transaction Server V3.1 provides new tools and capabilities that focus on three major areas to help CICS successfully implement new Web services applications. It is written by Mark Lillycrop, chief analyst at Arcati Research (mark@arcati.com).

For further information contact:
URL: www-931.ibm.com/bin/can_oneoff/can_oneoff_cosmetic_surveyload.cgi;jsessionid=00007DHv0dWGHCqqwjeWzY-MGo3:v36bl69j?pg=can_oneoff/cf1_nopre_cics2.html&tags=tactic,CICSE2MAR05ICCR2,vc,cics2_ccr2,url,,typ,typ1_cics2.html,seg,105AE11E.
Or: www.arcati.com/cics.pdf.

* * *