

CICS\TS and SOS Conditions

June 23, 2006

By: Eugene S. Hudders

CICS/TS Performance Tuning Using C\TREK Course

This seminar covers a lot of tuning material and will require 8 hours a day. The total time for the seminar is 40 hours.

The followings topics included in this course are:

- Introduction to CICS Performance Tuning
- Using Operating System Information to Tune CICS/TS
- Tuning CICS/TS Processor Cycles
- Tuning Real Storage in CICS/TS
- Tuning Virtual Storage in CICS/TS
- Tuning CICS/TS Transaction Controls
- Tuning On Line VSAM Files
- Tuning NSR Files in CICS/TS
- Tuning CICS/TS LSR Buffer
- Tuning the Index CISZ
- Reviewing VSAM Free Space
- Reviewing the DB2 Interface

If you wish to attend this course, please send us an email at ctrek@actpr.com

Since its inception in 1969 as an IBM product, CICS performance and reliability has always been sensitive to the availability of resources such as CPU processor capacity, I/O response and availability of storage, both real and virtual. However, of all the resources, the availability of virtual storage is probably the most important and can be referred to as the “Achilles Heel” of CICS. The lack of virtual storage can result in system outage or the infamous Short on Storage (SOS) condition. CICS outages due to a lack of virtual storage result from an operating system type GETMAIN failure. SOS conditions can occur due to lack of virtual storage in the CICS dynamic storage areas either above or below the line. Over the years CICS has been enhanced and re-architected with some major improvements being made in the way CICS handles virtual storage. This article addresses the improvements made and what can be done to resolve virtual storage problems that affect CICS/TS performance, reliability and availability.

CICS/TS runs within an address space of the z/OS operating system. The maximum storage currently available to CICS/TS is 2 GB or 31-bit addressing. Although 64-bit addressing has been announced, the CICS implementation of this feature has not been announced yet. The amount of virtual storage available for CICS processing is determined by the amount of storage made available when CICS is initialized. The amount of storage assigned can be provided by an installation default, by the coding of the REGION parameter in the start-up JCL or by the SMF installation exit IEFUSI. The most common way is to specify a particular size using the REGION parameter. However, most installations have the SMF exit active to control the amount of virtual storage that a job can request. It is recommended that CICS regions be allowed to allocate the maximum size of 2 GB by coding REGION=0M in the CICS start-up

JCL. The potential 2 GB allocation gives the CICS system programmer the capability to dynamically increase the amount of virtual storage up to 2 GB in the case of an SOS condition. In addition, having sufficient virtual storage available allows the CICS system programmer to add resources as a result of normal installation growth or new applications. This extra virtual storage avoids problems when additional resources have been added and the system fails to

initialize because of lack of virtual storage. **Figure 1** provides the maximum virtual storage available above and below the line after the operating system (common areas) have been reduced (Green Arrow).

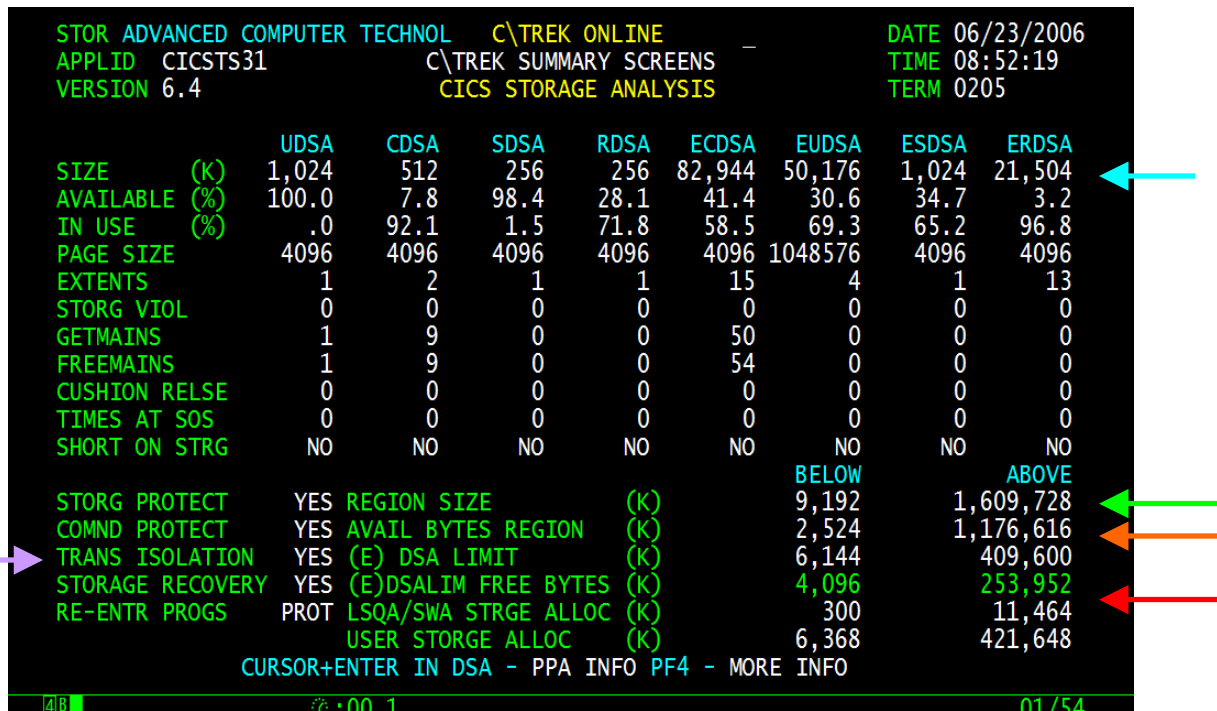


Figure 1. Virtual Storage Summary

Many installations use IEFUSI to reduce the amount of storage allocated by active jobs in the system. The reasons for these controls are that the system-wide allocation of paging data sets space is usually much lower than 2 GB per potential address space running in the system and that there are some programs (e.g., compilers and sort) that will use the total amount of available storage. Running short of slots on the paging data sets affects the entire system and can cause a system slowdown or system outage. CICS will use only what is required for operation and any unneeded virtual storage is simply not used. However, should the need arise that CICS requires additional virtual storage to eliminate SOS conditions above the line and/or add additional buffers or resources to improve response times or new functions, then having provided CICS the maximum space possible allows the system programmer to make these changes without having to recycle CICS with a larger REGION size. It is not unusual for the CICS system programmer to find out about region size limitations set by the IEFUSI routine at the time when there is a need to increase the CICS region size. Delays as a result of discovering the IEFUSI restrictions can be very exasperating.

There is another important use for the IEFUSI routine that does not necessarily relate to the control the maximum amount of virtual storage allocated. In recent years IBM has enhanced CICS to use more processors when dispatching work. In order to dispatch work and be able to better utilize multi-processing systems, CICS needs to use more operating system Task Control

Blocks or TCBs. Through the use of L8 and L9 TCBs, CICS can now dispatch threadsafe application program to run at the same time as another application program is running on the QR TCB. TCBs are allocated below the line in the LSQA. The amount of storage required to run CICS/TS is greater than the 16 MB available below the line. During region initialization, the operating system reserves a certain amount of virtual storage at the high end of the region below the line to store control blocks (e.g., TCBs) needed for CICS execution. The most common areas in the region below the line are the LSQA, the Scheduler Work Area (SWA), the Subpools 229/230 areas and the Authorized User Key (AUK) areas. At CICS initialization z/OS estimates the amount of storage required to support these areas. However, if the CICS system requires more storage (e.g., a large number of L8/L9 and J8/J9 TCBs) and no virtual storage is available, then the CICS initialization or execution is terminated. The only way a user can increase the amount of storage available for these operating system areas is through the IEFUSI routine that can be used to lower the available region size below the line leaving a larger area for the operating system control blocks.

An important figure that has to be determined quickly when CICS region is operational is the total amount of virtual storage allocated to CICS and the total amount of virtual storage that is available to increase resource definitions used by CICS. Knowing the amount of available region storage allows the CICS system programmer to increase the resources available to CICS without causing a GETMAIN failure or not being able to open the resource such as an LSR pool or a file. The information desired is simply the total amount of virtual storage available above or below the line minus that part required for CICS and operating system information. **Figure 1** provides the information regarding the amount of virtual storage available above and below the line for CICS use in the region (Orange Arrow).

There are many CICS requirements for virtual storage within a region with the some of them being the dynamic storage areas above (EDSA) and below (DSA) the line, VSAM buffers, file control blocks, trace table and programs (CICS, third party, Language Environment, etc.). Once CICS is loaded, the amount of virtual storage remaining is available for use such as opening additional files that may require buffer areas to be built or increasing the trace table size. An important use of the available region storage is to improve the CICS response time by adding additional buffers to the LSR pools to obtain a better look-aside hit ratio. It is imperative that at least 250 to 300 KB of virtual storage be left below the line in case z/OS has to obtain control blocks (i.e., SDWA) to capture a dump in the event of a CICS failure. Therefore, having "MVS" storage available within the region after CICS comes up is an important consideration to avoid any possible GETMAIN failures (e.g., S80A or S40D). Excess operating system ("MVS") region storage is not used by CICS, is not GETMAINed and does not require any backing on the page data sets.

One of the major CICS tuning areas are the dynamic storage areas above and below the line. These areas are used by CICS to hold user areas and programs as well as CICS tables and control blocks. The amount of storage allocated to these dynamic storage areas is determined

by the SIT parameters DSALIM for the DSA below the line and EDSALIM for the EDSA above the line. The amount of dynamic storage required will vary by CICS system and will depend on several factors such as transaction activity and volume, amount of resources defined (e.g., table definitions), the number of non-resident programs loaded during the normal CICS execution, Temporary Storage MAIN requirements among other things. The important objectives when selecting the (E) DSA values is not to be meek, that is, do not be afraid to allocate more than you need. CICS will only allocate what is required to support the applications running. For example, suppose you allocated an EDSALIM of 400 MB and you only required 150 MB. CICS will only allocate the required amount of 150 MB to support what the applications require and the remaining 250 MB will not be used. **Figure 1** provides the amount of unused (E) DSA storage still available for extents (Red Arrow).

The dynamic storage areas are subdivided into eight different pools. There are four pools below the line and four similar function pools above the line. The pools are used to hold different types of storage areas used during execution. The pool usage is as follows:

- (E) UDSA – the User DSA that is used to hold user data areas such as working storage and associated application areas
- (E) CDSA – the CICS DSA that is used to hold CICS control blocks and tables as well as CICS key programs
- (E) RDSA – Read-Only DSA that is used to hold re-entrant programs
- (E) SDSA – Shared DSA that is used to hold non re-entrant programs and GETMAIN SHARED storage areas

These pools are dynamically allocated as needed out of the DSALIM and EDSALIM allocation. **Figure 1** displays the current virtual storage use by (E) DSA as well as other information (storage violations, SOS, amount available, etc.) that relates to the health of the pool (Light Blue Arrow).

CICS ensures that the amount of virtual storage required by the DSALIM and EDSALIM is available and will reserve this amount out of the region storage. However, the unused portion will not be used but will be available should CICS require additional space. CICS allocates (E) DSA storage in increments called “extents” out of the total amount allocated in the DSALIM and EDSALIM SIT parameters. DSA extents below the line are allocated in 256 KB increments except if Transaction Isolation is active. In this case, the UDSA extents are 1 MB in size and acquired on a one MB boundary. The other DSAs (CDSA/RDSA/SDSA) are still allocated in 256 KB increments even when Transaction Isolation is specified. There is an exposure for virtual storage fragmentation below the line that may result in SOS conditions as a result of the mixed extent size and boundary requirements for allocations below the line (256 KB and 1 MB) when Transaction Isolation is active. EDSA extents above the line are allocated in 1 MB increments. If the storage request is greater than one extent, then multiple extents can be acquired at one time as one contiguous piece of storage to satisfy the request.

Virtual storage is allocated and released during CICS transaction execution from the (E) DSAs. When a request for virtual storage from a particular (E) DSA exceeds the amount of storage available in the extent(s) from a particular (E) DSA, a new extent is acquired to satisfy the request, if unused virtual storage is still available from the initial DSALIM or EDSALIM allocation. This process continues until the remaining unused (E) DSALIM virtual storage is exhausted. It is at this point when CICS is more prone to problems associated with lack of virtual storage.

During normal CICS execution, the Storage Manager Domain (SM) monitors the storage in the (E) DSAs. CICS uses a dynamic program storage compression algorithm. The SM Domain works in conjunction with the Loader Domain (LD) in ensuring that unneeded programs are removed from storage when a certain storage objective is reached. Each executing program has a use count that identifies the current number of tasks using the program. This counter is incremented by one each time a program is used. Once a transaction finishes using a program, the current use count is reduced by one. Once an executing program is not in use (current use is equal to zero), the LD maintains the program in the storage in order to reduce the overhead of program loading from the DFHRPL. The unused program is placed on a list called a Not-In-Use (NIU) queue and the size of the program is considered when computing a total called Redundant Program Storage (RPS). The programs in the NIU queue are kept in a Least Recently Used (LRU) order, that is, the oldest unused program is at the bottom of the list. If a program that is in the NIU queue is needed, it is reused and taken off the NIU queue and not considered in the RPS total. Once the program is reused and no longer needed, the program is again returned to the top of the NIU queue and included in the RPS total.

The amount of RPS in each (E) DSA is reviewed to ensure that it does not exceed a certain target. An RPS target factor of 50% is used in all (E) DSAs. The target percentage is kept in the LD anchor block (Figure 2). The user cannot alter this factor. Whenever the use count of a reusable program goes to zero, the amount of storage it occupies is used to compute the RPS figure for the appropriate (E) DSA. If the RPS for the (E) DSA exceeds the computed target, then LD will delete as many not-in-use programs as is required from the (E) DSA concerned to make the RPS less than the target. The programs are deleted on a "Least Recently Used" basis. The number of programs deleted depends on the figure that is required to lower the amount of virtual storage below the target value. This deletion process is called program compression although no compression is actually done because programs remain at their assigned load point address.

The system does not provide a statistic as to the number of times program compression has occurred. However, there is an indirect indication that program compression has occurred. This indirect measurement is the number of times programs have been fetched from the DFHRPL library. Programs may have been fetched because a new copy was issued. Program fetching not related to new copies is a good indication that the DSALIM and/or EDSALIM are

small and may require adjustment. In addition, program fetching is usually the first warning a user gets that SOS conditions are just “around the corner”. Program fetching above the line can usually be fixed by simply increasing the EDSALIM size. Naturally, you could run into some problems when increasing the EDSA if you do not have sufficient operating system virtual storage available to handle the increase request because you did not specify REGION=0M or a large region size.

```

LD2  ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE      1E012000  DATE 06/23/2006
APPLID CICSTS31                  LOADER DOMAIN      TIME 09:15:52
VERSION 6.4                      LOADER STATISTICS  TERM 0205

STATS POINTER 1E012390 # PGMS DEFINED      2264 # PGMS DELETED      0
# PGMS INQUIRES 0 # PGM REFRESHES  29 # START BROWSES   0
# SM NOTIFIES   0 # DFHRPL LOADS    0 # PGM REUSED       0
# CURR SUSPENDED 0 TOTAL # SUSPEND  0 HWM SUSPENDED      0
# HWM ATTAINED  0 # DEB REBUILDS   0 STORAGE FACTOR    50
# NAME LONG CACHE 0 LEN LONGEST NAME 0 ADDS TO CACHE     0

                                LOADER TIME STATISTICS
TOTAL LOAD TIME 00:00:00.0      AVG LOAD TIME 00:00:00.0
TOTAL WAIT TIME 00:00:00.0      AVG WAIT TIME 00:00:00.0
LAST RESET TIME 09:00:00.1

DSA-NAME  STOR-USE  RPS-STOR  STR-TRGT  PGM-REMO  PGM-RCLM  NIU-SIZE  NIU-Q-TIME
CDSA      27         0          16         0          0          0 00:00:00.0
SDSA     15645      17        2048        0          0          6 00:00:00.0
RDSA      0         0          16         0          0          0 00:00:00.0
ECDSA     34         34        2048        0          0          4 00:00:00.0
ESDSA     170        0          16         0          0          1 00:00:00.0
ERDSA    20759      602        2048        0          0          45 00:00:00.0
ENTER REFRESH PF1 HELP PF3 PREV PAGE PF5 MEMORY CLR MAIN MENU
4B 00.1 01/54
    
```

Figure 2. Loader Domain Anchor

Program fetching below the line may be caused by many factors, one of which may be fixed by increasing the DSALIM size, if virtual storage below the line is available. However, due to the low availability of virtual storage below the line which is usually in the range of 6 to 9 MB, the SOS conditions may be due to applications still running below the line. SOS conditions in the UDSA may also be possible even with virtual storage availability from the DSALIM because of the use of Transaction Isolation [Figure 1](#) (Lilac Arrow). The problem may be caused by fragmentation because any additional UDSA extents have to be allocated in 1 MB increments and must be on a one MB boundary. Since other DSAs below the line are allocated in 256 KB increments, you could wind up with available virtual storage but not necessarily contiguous or cannot be allocated on a one MB boundary aligned extent. [Figure 3](#) provides a glimpse of the fragmentation that can occur in the DSA. The Red Arrow shows an empty extent of 768 KB followed by a 1 MB allocation for the UDSA. As the UDSA extent had to be on a one MB boundary, the 768 KB was bypassed. This area is available for extents from the other DSAs below the line. The Green Arrow in [Figure 3](#) provides the remaining free storage from the

DSALIM available for extents. The sum of the two free areas is 4 MB but it consists of two fragments.

```

DXH  ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE      1DF27250  DATE 06/23/2006
APPLID CICSTS31                STORAGE MANAGER DOMAIN  TIME 09:26:56
VERSION 6.4                    MVS STORAGE MAP        TERM 0205

DXG-NEXT DXG-PREV LOW-ADDR HIGH-ADR  LENGTH KEY  SP  PPX-ADDR DSA-NAME  FLG
1DF272A0 1DD662E8 00040000 0007FFFF 256 80 130 1DE9B040 CDSA 00
1DF27480 1DF27250 00080000 000BFFFF 256 00 252 1DE9B9B8 RDSA 00
1DF27750 1DF272A0 000C0000 000FFFFF 256 90 130 1DFEC350 SDSA 00
1DF27020 1DF27480 00100000 0013FFFF 256 80 130 1DF4F068 CDSA 00
1DF277F0 1DF27750 00140000 001FFFFF 768 00 129 00000000 00
1DF277C8 1DF27020 00200000 002FFFFF 1024 90 130 1DF7F588 UDSA 00
1DD662E8 1DF277F0 00300000 0063FFFF 3328 00 129 00000000 00
1DF27110 1DD66360 1E000000 1E0FFFFF 1024 80 130 1DE9ACF0 ECDSA 1E
1DF27160 1DF270C0 1E100000 1E1FFFFF 1024 00 252 1DE9A808 ERDSA 1E
1DF271B0 1DF27110 1E200000 1E2FFFFF 1024 00 252 1DE9A6B8 ERDSA 1E
1DF27200 1DF27160 1E300000 1E3FFFFF 1024 00 252 1DE9A568 ERDSA 1E
1DF272F0 1DF271B0 1E400000 1E4FFFFF 1024 00 252 1DE9A418 ERDSA 1E
1DF27340 1DF27200 1E500000 1E5FFFFF 1024 00 252 1DE9B868 ERDSA 1E
1DF27390 1DF272F0 1E600000 1E7FFFFF 2048 00 252 1DFECA18 ERDSA 1E
1DF273E0 1DF27340 1E800000 1E8FFFFF 1024 00 252 1DEA1078 ERDSA 1E
GM STOR BELOW 6144 GM STOR ABOVE 409600
EXTENTS BELOW 7 EXTENTS ABOVE 35
ENTER REFRESH PF1 HELP MORE D
PF2 SEARCH PF3 PREV PAGE PF5 MEMORY PF7 BACKWARD PF8 FORWARD CLR MAIN MENU

```

Figure 3. CICS Extent Storage Allocation Map

In these cases, you may want to pre-allocate the amount of UDSA required at system initialization by using the SIT parameter UDSASZE. For example, if you determine that your UDSA requires 4 MB of storage to run with minimum or no SOS conditions, then you can ensure that CICS will allocate a contiguous 4 MB UDSA by using the UDSASZE parameter. Pre-allocating the UDSA avoids not being able to allocate an extra extent in the UDSA because of fragmentation or boundary alignment. However, you can specify the UDSASZE parameter only in PARM, SYSIN, or CONSOLE. It is important to note that if this parameter is used, CICS will no longer dynamically allocate extents for this DSA, even if more space is available.

During the normal execution of CICS, the SM monitors the availability of virtual storage to satisfy requests. Each (E) DSA has a storage cushion value that is generally 64 KB for DSAs below the line and 128 or 256 KB for EDSAs above the line. The EUDSA has a cushion value of 0 KB and the control mechanism is described later on in this article. You no longer can define the cushion size as you did in older versions of CICS. Whenever the amount of available virtual storage in a particular (E) DSA falls below this value, CICS will allocate a new extent if virtual storage is still available from the initial EDSALIM or DSALIM. Once all the available virtual storage has been allocated, CICS has an SOS condition. CICS handles the situation by requesting a program compression to release unused programs to see if sufficient virtual storage can be acquired to satisfy the request. If CICS can satisfy the request, then normal

processing continues. However, if in order to satisfy the request, CICS has to use some of the 64 KB from the storage cushion, then CICS will allow this but will signal that the storage cushion has been released. A statistic proving the number of times the cushion has been released is provided.

If the program compression is unable to resolve the SOS condition, then CICS issues the system under stress message (DFHSM0131 or DFHSM0133) and turns on the SOS flag in the system. CICS also provides a statistic indicating the number of times the SOS condition has occurred. Note that program compression can be an effective method for recovering virtual storage for the dynamic storage areas but there are no programs present in the UDSA or EUDSA. So, program compression does not directly help these two dynamic storage areas in relieving an SOS condition that occurs in these dynamic storage areas. The only alternative is that during program compression of another (E) DSA an extent becomes totally free and is available for stealing to be used by the UDSA or EUDSA. If no extent becomes available, then we are in for a “rough day at the ranch”. Once CICS turns on the SOS flag, then the entire task attachment mechanism is stopped. CICS stops accepting new transactions and tries to work with the tasks in the system. CICS performance and response times are affected. If the system cannot resolve the situation, you may have to re-cycle CICS to get the condition resolved. If access to a performance monitor can be made on a timely basis, you may be able to cancel sufficient tasks to resolve the condition and let CICS continue to process. Once the SOS condition is resolved, CICS issues the system is no longer under stress messages (DFHSM0132 or DFHSM0134).

The major problem with the system under stress message is that no information is produced so that you can determine the cause of the SOS condition. So, you could wind up with a series of system under stress followed by a system is no longer under stress messages and not have a clue as to what caused the problem nor what task or tasks are to blame. To debug this type of problem you would need to determine who is causing the situation and what is happening in the system. In order to get information, the user has to set the system up to cause a system dump when an SOS condition occurs. For example, you can request a dump by entering the master terminal command of “CEMT S SYD(SM0133) ADD SYS MAX(1)”. This will generate a system dump when an SOS condition occurs above the line. If the desired dump is below the line, simply change the dump code to SM0131. However, you will have to inspect this dump using IPCS offline. Unfortunately, there are installations that do not have the expertise to debug an IPCS dump. Your performance monitors may also not be of much help either although there are other tools that can provide this information (Figure 4). The Yellow Arrow identifies the (E) DSA that reflects the SOS condition while the Red Arrow identifies the transaction that caused the SOS condition and the amount of storage requested.

```

AHDD ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE          DATE 06/23/2006
APPLID CICSTS23                   MISCELLANEOUS      TIME 14:53:47
VERSION 6.3                       SV/ABEND HANDLER-  TERM 0205

CICSTS23 DSA-NAME  CURR-ALLOC  CURR-USED  CURR-FREE  SOS
CICSTS23 UDSA     00000001024 00000000000 00000001024 N
CICSTS23 CDSA     00000000512 00000000016 00000000072 N
CICSTS23 SDSA     00000000256 00000000000 00000000244 N
CICSTS23 RDSA     00000000256 00000000000 00000000080 N
CICSTS23 ECDSA    00000048128 00000017924 00000016864 Y ←
CICSTS23 EUDSA    00000017408 00000001024 00000016384 N
CICSTS23 ESDSA    00000002048 00000000000 00000001392 N
CICSTS23 ERDSA    00000019456 00000000000 00000001012 N
CICSTS23 DSALIM=000006144K          EDSALIM=000089088K
CICSTS23 TOTAL-BELOW=000002048K     TOTAL-ABOVE=000087040K
CICSTS23 TRAN=TREC TRAN#=00062 TERM=0202 PROG=KVPKVEW STOR=0000400K ECDSA ←
CICSTS23 C24A=0000000K C31A=0000400K U24A=0000000K U31A=0000000K 17384592
CICSTS23 C24U=0000000K C31U=0000392K U24U=0000000K U31U=0000000K
CICSTS23 TRAN=TSVR TRAN#=00043 TERM=          PROG=KVPTSVR STOR=0001080K
CICSTS23 C24A=0000000K C31A=0000056K U24A=0000000K U31A=0001024K
CICSTS23 C24U=0000000K C31U=0000048K U24U=0000000K U31U=0000103K
CICSTS23 TRAN=CEMT TRAN#=00061 TERM=0204 PROG=DFHEMTD STOR=0000032K
CICSTS23 C24A=0000004K C31A=0000028K U24A=0000000K U31A=0000000K
PF2 SEARCH PF7 BACKWARD PF8 FORWARD CRSR +ENTER POSITION LINE      MORE B
4B  :00.1 01/54
    
```

Figure 4. SOS Information

CICS has another protection mechanism because program compression may not help SOS conditions in the UDSA and/or EUDSA (Figure 4). CICS uses two other action items to try and defer SOS conditions related to these (E) DSAs. These action items vary in name but are based on a range of free virtual storage available in the UDSA and/or EUDSA. The first action occurs when the amount of free virtual storage lies between 128 to 256 KB (Figure 5 Yellow Arrow). At this point, new tasks are delayed for an interval of 50 milliseconds (Figure 5 Red Arrow). The idea behind the delay is to provide an opportunity for executing tasks to complete and release allocated virtual storage from the (E) DSA being used by the task. This delay also defers the new task from obtaining virtual storage for its work areas (e.g., WORKING-STORAGE) and program, if the program is not in memory at the time. The virtual storage check is sometimes called “Going SOS”. If the previous action does not solve the situation and the available virtual storage falls between 64 to 128 KB, then CICS delays the tasks even further. The delay interval is 500 milliseconds. The virtual storage range is sometimes called “Storage Critical”. The system enters SOS territory when the amount of virtual storage falls below 64 KB. In this case, all tasks attaches are stopped. The target virtual storage ranges and delays are kept in the Dispatcher anchor block. These values cannot be altered. There are no CICS statistics available to identify that the actions were taken. The UDSA (64 KB) and EUDSA (128 KB) cushion sizes are kept in the Transaction Manager Anchor (XM) block (Figure 6).

```

DS  ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE      1DF33000  DATE 06/23/2006
APPLID CICSTS31                 DISPATCHER DOMAIN    TIME 09:57:09
VERSION 6.4                     DISPATCHER ANCHOR BLOCK TERM 0205

NUMBER OF CO TCBS                01          QR TCB CPU USE %      .00
PRIORITY AGING FACTOR            32768       NUMBER OF TASKS:
ICV (MS)                         10000       CURRENT              21  PEAK      24
ICVTSD (MS)                      0           SHORT ON STORAGE (K BYTES):
SIT ICV (MS)                     10000       GOING                256  CRITICAL 128
SIT ICVTSD (MS)                   0           ACT HAND POSTING TCBS 0
HAND POST INTERVAL (MS)          100         MAX HAND POST TCBS   0
EVERY SO OFTEN SCAN (MS)        500         ACT JVM TCBS         0
NEW TASK DELAY                   50          MAX JVM TCBS         0
NEW TASK PENALTY                 500         ACT OPEN TCBS        20
MROBATCH                          1           MAX OPEN TCBS        20

CSA ADDRESS                      8004F0C0    CURRENT TIME          09:57:08.9
EXECUTE CHAIN ADDR              1DFF8200    NEXT EXEC TIME        09:57:09.4
DELAY CHAIN ADDRESS             00000000    HAND POST CHAIN TIME  09:57:09.0
DEAD TCB POOL ADDRESS           00000000    DEADLOCK TIMEOUT GAP  500
HAND POST CHAIN:
  START 00000000  END 00000000  NEXT DEADLOCK TIMEOUT 00:00:00.0
NEXT TIMER EVENT                09:57:12.6
NEXT TCP DISPATCH TIME          09:57:18.9

PF5 MEMORY PF6 DS STATS PF8 MORE INFO PF9 TASK LIST PF10 TCB LIST PF11 TCB STAT
  
```

Figure 5. Dispatcher Controls for SOS

```

XM  ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE      1E004000  DATE 07/06/2006
APPLID CICSTS31                 TRANSACTION MANAGER DOMAIN TIME 10:33:47
VERSION 6.4                     TRANSACTION ANCHOR BLOCK  TERM 0210

XM ANCHOR ADDRESS              1E004000    LOCAL SYSID           C310
1ST XMT ON CHAIN              1E0091B8    NUMBER OF DETACHES    72936
LAST XMT ON CHAIN             1E00AC70    NUMBER OF ATTACHES    12
NEXT TRANS NUMBER              73082      SYSTEM AT MAX TASKS?  NO
NUMBER OF TCLASSES            16          24-BIT CUSHION (K)    64
1ST TCLASS ADDRESS            1E00C730    31-BIT CUSHION (K)    128
LAST TCLASS ADDRESS           1E00C7B0    LAST XM RESET TIME    09:00:00.1
MAXIMUM TRANS RANGE           99999      MAX TASK SYSTEM VALUE 100
MAX TASK CHAIN ADDR           1E00C030

XM TRANSACTION DIRECTORY ADDRESSES

TRAN DIRECTORY ADDR           1E000040    NUMBER OF INSTANCES CRTD 238
REMOTE TRANS DIR ADDR         1E0005A0    DTR TRANSACTION ID      CRTX
TPN NAME DIR ADDRESS          1E000B00    TRANS DEF INST ADDRESS  1F02D030
TRANS DIR LOCK ADDR           1DFF3E28    VALIDATION NUMBER       3A

ENTER REFRESH PF1 HELP
PF3 PREV PAGE PF4 TRANS CHAIN PF5 MEMORY PF6 CLASS MAX TASKS CLR MAIN MENU
  
```

Figure 6. SOS Cushion Values UDSA/EUDSA

There are several things that you can do to help CICS resolve an SOS condition without human intervention. The use of DTIMOUT and SPURGE in the transaction definitions provides CICS with a tool whenever an SOS condition occurs. If a deadlock condition occurs because CICS cannot dispatch any tasks because of the SOS condition, CICS can start to purge transactions that have reached their DTIMOUT value and have SPURGE specified. This way, CICS can try to resolve the situation without someone intervening. It is important that all query, menu and non-essential transactions have these parameters specified to allow CICS to try and correct the situation. Other transaction types can also be included but that is an installation decision. The time out value should be set sufficiently low (e.g., one to five seconds) to avoid having to wait long periods of time before action can be taken.

As we discussed previously, there should be no SOS conditions above the line because of the amount of virtual storage available above the line. Baring a runaway task that continuously requests storage in a loop, SOS conditions above the line can be fixed by increasing the EDSALIM. Therefore, ensure that a large EDSALIM is specified for the EDSA. Fixing an SOS problem below the line is more challenging. Increasing the DSALIM to fix the problem may not be easy because of the limited amount of virtual storage available below the line. Naturally, the best solution is to convert the applications running below the line to run above the line but this may not be an easy task nor can it be done overnight to resolve current SOS conditions. If the migration of the application below the line is not possible, then there are other options available but should be considered temporary, especially if the transaction volume below the line is increasing. The following paragraphs describe some of these options for you. The degree of effort required to implement some of these options will depend on the information that you have available. Also, you may or may not be able to use some of these alternatives.

A common solution to virtual storage problems is simply to clone another CICS region. This alternative is relatively easy if multiple applications are running in the region and all that is required is to split the applications into separate regions. However, if there is just one application or the applications are inter-related, then the split may not be as easy. It may involve analyzing the application for inter-dependencies, correcting them and then splitting the application to be serviced by two regions using a dynamic routing routine. If the files are inter-related, it may require creating a File Owning Region (FOR). In addition, you may need to consider the amount of virtual and real storage required by a new CICS.

One of the first things that can be done is to take all of the BMS maps/MAPSETS and re-link them above the line. BMS will handle maps above the line even though your application runs below the line. This measure may release sufficient storage so that either your SDSA (maps defined as programs) or your CDSA (MAPSETS) can release an extent that can be used by other DSAs. Another option depends on the use of the TCTUA below the line. Many installations simply code a TCTUA length of 255 for all/most terminal definitions even though the programs may use less or none of the TCTUA assigned. If you could determine the highest referenced TCTUA position, then you could determine if you could lower the allocation. For

example, imagine that you measure the highest TCTUA position over a certain period of time and determine that the highest position used was 69 (Figure 7). If you have the TCTUA set to 255 bytes and have 2000 active terminals, then you would require 510,000 bytes (255 X 2000) below the line to support the TCTUA structure when you are using a maximum of 138,000 bytes (69 X 2000). That is an unproductive use of virtual storage below the line. The 510 KB TCTUA allocation is almost the equivalent of two DSA extents. If you lower the TCTUA size from 255 to 127, you would be able to save 256 KB (128 X 2000) or one “extent” below the line that would be available for productive use. The best result for this option is in a Terminal Owning Region (TOR) but you have to remember that the TCTUA is passed to the Application Owning Regions (AOR).

```

TCTU ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE          DATE 06/23/2006
APPLID CICSTS13                  APPLICATION DOMAIN    TIME 10:04:36
VERSION 5.3                      TCTUA STATISTICS    TERM 0205

TOTAL TERMINALS                   6
TCTUA TERMINALS                   2    33.33 % OF TOTAL TERMINALS
NON-TCTUA TERMINALS               4    66.66 % OF TOTAL TERMINALS
UNUSED TCTUA                      0    0.00 % OF TCTUA TERMINALS
TOTAL TCTUA BYTES                 510
AVERAGE NUM OF BYTES             255
MAXIMUM SIZE                      255
MINIMUM SIZE                      255
TOTAL BYTES UNUSED                 0    0.00 % OF UNUSED STORAGE ←
TCTUA KEY                         USER
TCTUA LOCATION                    BLW

HIGH USED TCTUA POSITION            69  (1-255) (0 = TCTUAs ARE UNUSED)
TOTAL POSSIBLE FREE BYTES          372
POSSIBLE PERCENT SAVED             72.94

ENTER REFRESH PF1 HELP PF3 PREV PAGE PF5 MEMORY CLR MAIN MENU
4B  :00.1 01/54
    
```

Figure 7. TCTUA HWM Usage

Another alternative is to locate unused Transaction Work Areas (TWA) for applications that run below the line. It is not unusual for programmers that converted Macro Level applications to Command Level to have moved the TWA definition from the Linkage Section in the COBOL program to the WORKING-STORAGE. However, if the application programmer failed to inform the CICS system programmer of the change, then the transaction definition is still requesting the TWA size during execution and remains unused during transaction execution wasting virtual storage. Locating this type of condition may involve a meeting with the programmers involved in the conversion, a review of the program listings or a tool that performs this analysis (Figure 8). Naturally, this example shows that CSTP does not use its TWA but you would not want to eliminate the TWA for CSTP without IBM’s consent. All identified transactions should be reviewed by the application staff to ensure that the TWA is not needed.

```

TWA ADVANCED COMPUTER TECHNOL C\TREK ON-LINE 1E004000 DATE 06/23/2006
APPLID CICSTS31 TRANSACTION MANAGER DOMAIN TIME 11:08:03
VERSION 6.4 UNUSED TWA TERM 0205

TRANS TRAN PTY ATTACH TCA TWA TWA ALLOC USED ALLOC USED
NUMBER ID TIME ADDRESS ADDRESS SIZE BELOW ABOVE BELOW ABOVE
      8 CSTP FF 14:53:07.4 1E0BA080 1EE7D468 396 0 0 4 2
    
```

←

PF7 BACKWARD PF8 FORWARD PF12 MORE PFKEYS OPTIONS

4E:00.101/54

Figure 8. Unused TWAs

Users can also control the number of transactions that can enter the system for execution below the line by using the transaction class (TCLASS) mechanism. TCLASS allows the user to control the total number of transactions that can enter the system for execution below the line. MXT is a global control which cannot be used to select specific transactions that execute below the line. Through the use of TCLASS, the user can determine the right number of transactions that can be serviced below the line without over committing the amount of virtual storage available below the line. The maximum associated with the TCLASS can be monitored and modified via a CEMT command. In addition, a maximum threshold value can be specified that can be used to automatically cancel queued transactions when certain levels of transactions are waiting for a particular TCLASS. Assigning transactions to a TCLASS can be tedious and time consuming when there is a long list of transactions that run below the line. However, the results can be very effective in controlling SOS conditions.

If the SOS condition is below the line, then the Language Environment (LE) options (CEEEOPT) should be reviewed to ensure that the virtual storage parameters (e.g., Stack, Heaps etc.) are set correctly for a CICS environment. The batch LE options are not good for a CICS environment. In addition, you must ensure that secondary allocations take the Storage Check Zone (SCZ) requirements into consideration. For example, a secondary allocation of 4096 would result in 8192 bytes being acquired because CICS requires that the acquired area be enclosed by two SCZs. An SCZ is 8 bytes long and you would need two SCZs (16 bytes) to enclose the 4096 area resulting in more than one page being acquired (Figure 9).

```

CEEC ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE      000ACCC0  DATE 06/23/2006
APPLID CICSTS31                   APPLICATION DOMAIN  TIME 14:51:11
VERSION 6.4                       RUN TIME OPTIONS  TERM 0205
                                LE SELECTED CICS OPTIONS

OPTION  STAT OVR  INITIAL  INCREMNT  ANYWHERE  KEEP  INITSZ24  INCRSZ24
        ON  OVR
ALL31   ON  OVR
ANYHEAP ON  OVR      4096      4080  ANYWHERE  FREE
BELOWHEAP ON  OVR      4096      4080  BELOW     FREE
HEAP    ON  OVR      4096      4080  ANYWHERE  KEEP      4096      4080
HEAPCHK OFF  OVR         1         0
LIBSTACK ON  OVR        32        4080  BELOW     FREE
RPTOPTS OFF  NVR
RPTSTG  OFF  OVR
STACK   ON  OVR      4096      4080  ANYWHERE  KEEP
CHECK   ON  OVR      NONE      NONE    NONE      00000000
TRMTHDACT ON  OVR  TRACE
CBLPSHPOP ON

STORAGE          HEAP          HEAP          DSA          SOS
                  INIT VALUE  FREE VALUE  INIT VALUE  RESERVE SIZE
                  ON  OVR      NONE          NONE          NONE          0

ENTER REFRESH PF1 HELP PF3 PREV PAGE PF6 RCMNDTN PF9 ALL OPTNS PF11 ROPT/COPT
4B :00.4 01/54
    
```

Figure 9. LE CEECOPT Options

One final option is to analyze the amount of virtual storage used by task. This alternative is mentioned but not many installations have the resources or time to dedicate to this option. You need to have information regarding what tasks were in storage when the SOS occurred to perform this type of analysis. This information may be obtained by reviewing a dump when the condition occurred or by data provided by a performance monitor. Depending on the depth of the analysis, the effort required to perform this task can be large because it may involve a detailed review of one or more programs. Therefore, before going into a more in depth analysis, you should try to determine if the problem is caused by a(n) (E) DSA that is too small for the task volume that is being handled. If this is the case, then other alternatives previously mentioned should be tried to resolve the situation.

There are basically two types of storage use in which you may want to concentrate the analysis. The first is simply identifying a task that has acquired a significant amount of storage. This type of a task may be in some type of loop in which storage is being acquired but not released. This is usually a program bug. However, what if there is no “blazing gun”, that is there is no major user of storage that can be easily identified? This is the second type of analysis. In this case, the amount of storage acquired by every task in the system has to be reviewed. You are interested in studying the total amount of storage requested and the actual amount used by each task (Figure 10). If the amount of used storage versus the percentage of the amount of storage used versus the amount allocated is a low percentage (e.g., 30%), then the programs associated with the task have to be analyzed to determine why this storage is acquired and the program modified to improve the situation. This information can be obtained from your on-line

performance monitor or by reviewing SMF records offline. Be aware that Transaction Isolation can yield some unusual results for tasks executing above the line because each task is assigned a minimum of one MB whether or not the task is going to use it.

```

XML  ADVANCED COMPUTER TECHNOL  C\TREK ON-LINE  1E704000  DATE 06/23/2006
APPLID  CICSTS23                TRANSACTION MANAGER DOMAIN  TIME 15:04:27
VERSION 6.3                    TRANSACTION CHAIN DISPLAY   TERM 0205

```

TRANS NUMBER	TRAN ID	PTY	ATTACH TIME	TCA ADDRESS	TWA ADDRESS	TWA SIZE	ALLOC BELOW	USED BELOW	ALLOC ABOVE	USED ABOVE
4	CSOL	FE	14:52:42.4	1E7B5680	NO TWA	0	0	0	4	2
5	CSSY	FF	14:52:49.9	0005D080	NO TWA	0	4	2	0	0
6	CSSY	FF	14:52:49.9	0005D680	NO TWA	0	4	2	0	0
8	CSTP	FF	14:52:56.4	1E7B6080	1F36E468	396	0	0	4	2
20	CFQS	1	14:53:30.7	1E7B7080	NO TWA	0	0	0	4	2
21	CFQR	1	14:53:30.7	1E7B7680	NO TWA	0	0	0	4	2
22	CSZI	1	14:55:14.2	1E7B6680	NO TWA	0	0	0	4	2
25	CSHQ	FE	14:56:38.0	1E7B8080	NO TWA	0	0	0	4	2
28	CSKL	FF	14:56:38.4	1E7B9680	NO TWA	0	0	0	20	20
33	CSNE	FF	14:56:44.7	1E7B9080	NO TWA	0	0	0	4	4
35	TSVR	FF	14:58:20.8	1E7B8680	NO TWA	0	0	0	1080	146
43	CSNC	FF	15:40:43.8	0005C680	NO TWA	0	4	2	0	0
203	TREC	FF	14:53:20.6	1E7BA080	NO TWA	0	0	0	15888	15877

PF7 BACKWARD PF8 FORWARD PF12 MORE PFKEYS OPTIONS

4B :00.1 01/54

Figure 10. Storage Use by Transaction

This article has covered many areas related to how CICS handles virtual storage and what preventive measures CICS uses to avoid SOS. In addition, the article reviewed some of the possible recommendations on how to resolve SOS conditions.

If you have any questions, comments or request for more information, please contact us:

Address: C\TREK Corporation PO BOX 560069 MONTVERDE FL 34756	Phone: (407) 469-3600 C\TREK Corporation (787) 756-5620 Advanced Computer Technology
E-mail: ctrek@actpr.com	Fax: (787) 756-5150
Website: www.ctrekcorp.com	Support: (787) 397-4150 (321) 297-5838 (787) 462-0406