Virtual Storage Analysis:
Exploring Below the Bar Private Storage Problems

NY Metro NaSPA - April 22nd, 2020

Patty Little plittle@us.ibm.com

Trademarks

SHARE
EDUCATE • NETWORK • INFLUENCE

The following are trademarks of the International Business Machines Corporation in the United States and/or other countries.

- **MVS™**
- **OS/390®**
- **z/Architecture®**
- **z/OS®**
- **OMEGAMON®**

* Registered trademarks of IBM Corporation

Table of Contents

Mention SWA and System Region here. Briefly talk about unauth/auth?

VIRTUAL STORAGE MANAGEMENT CONCEPTS

What is VSM?

SHARE
EDUCATE • NETWORK • INFLUENCE

VSM – Virtual Storage Manager

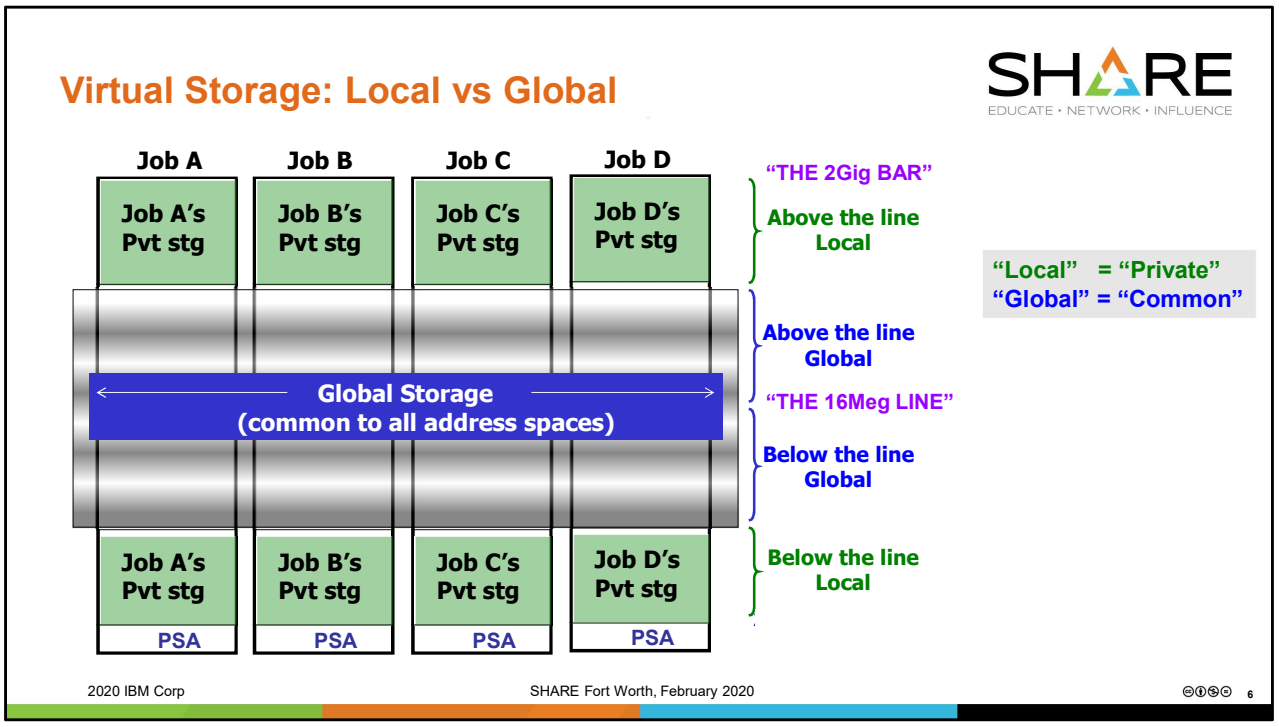
- Manages virtual storage below the 2Gig bar*
- Supports GETMAIN/FREEMAIN and STORAGE OBTAIN/RELEASE requests
- Issues ABEND878/ABEND80A to indicate "out of virtual storage" conditions
  - This presentation will focus on RC0C and RC10, both of which indicate "out of local virtual storage" conditions

* RSM manages virtual storage above the bar.

Virtual Storage Manager (VSM) is the component that manages virtual storage below the 2 Gigabyte bar. Upon going to z/Architecture, Real Storage Manager (RSM) took responsibility for managing above the bar virtual storage.

Functions running on z/OS can obtain virtual storage below the bar through the use of the GETMAIN macro or the STORAGE OBTAIN macro. Storage is returned to the system via the FREEMAIN macro or STORAGE RELEASE macro.

Exhaustion of an area of virtual storage can lead to virtual storage abends and the need to determine the culprit behind the storage misuse. Out of storage conditions in VSM are presented as either an ABEND878 or ABEND80A, with an accompanying return code indicating the area of storage that is exhausted. This presentation will provide instruction on how to diagnose virtual storage shortages.



The Local (private) areas of virtual storage are private to the owning Address Space (Job). Addressability to local storage is controlled by the owning address space, and the storage is not readily addressable from any other address space.

Programs and control blocks that live in global storage can be accessed by all jobs.

Private = Local; Common = Global.

Types of local storage

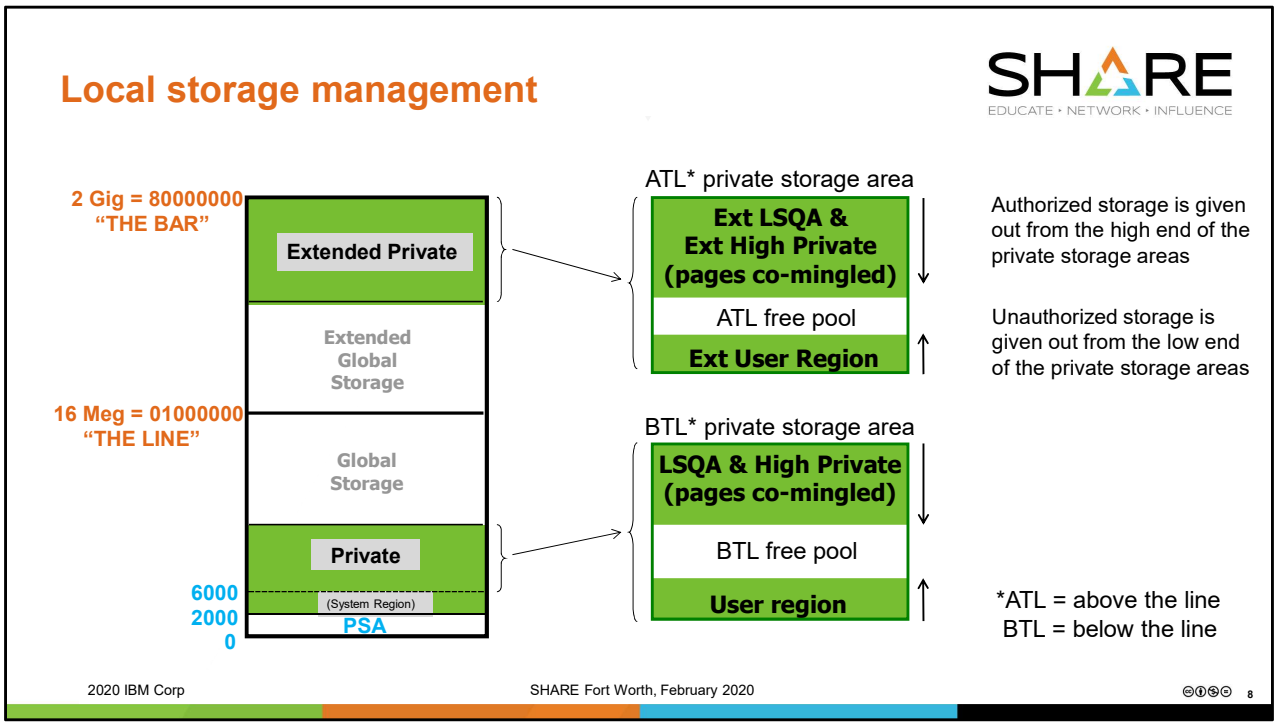
SHARE
EDUCATE • NETWORK • INFLUENCE

There are 3 main types of local storage

- Authorized storage (only obtainable by authorized code, e.g. DB2, BCP)
  - Local System Queue Area (LSQA)
  - High private
- Unauthorized storage
  - User region

SWA (System Work Area) is another type of authorized local storage, but it is highly specialized and will not be highlighted in this presentation.

The System Region is a specialized part of the User Region, for use only by the Region Control Task (first TCB in each address space). It occupies address range x'2000' thru x'5FFF' in every address space except ASID 1. System Region in ASID 1 also begins at x'2000' but is approximately x'30000' bytes in size. Discussion of System Region is beyond the scope of this presentation.



Each address space has a private storage area below the line and a private storage area above the line. The above the line area is referred to as the extended private storage area. The private storage areas are the same size for every address space on the system, although we will see that job-specific parameters (e.g. REGION parameter on the JCL) allow control of how much of the address space's private storage area is available for use by the application running within that address space.

Both private storage areas within an address space break down similarly into an area for unauthorized storage at the low end of the private storage box, and an area for authorized storage at the high end of the private storage box. The unauthorized area is known as the user region. The authorized area is used for LSQA and high private storage (as well as SWA). Unauthorized storage is given out starting from the low end of the private storage area. It grows up. Authorized storage is given out starting from the high end of the private storage area. It grows down. If the two areas grow together, or "bump", this results in an out of local storage condition. Another way to encounter an out of local storage condition is to have the user region hit the maximum allowed by a particular job, as defined by SMFLIMxx, REGION, IEFUSI, etc.

Note that pages of authorized storage cannot intermingle with pages of unauthorized storage. All pages below a particular point in the private storage area will be exclusively unauthorized user region (or free pages), and all pages above that point will be exclusively authorized storage (or free pages).

Local storage management implications

- Excessive growth of user region can affect LSQA/high pvt
  - System provides capability of limiting user region growth on a job by job basis
    - SMFLIMxx (see speaker notes for reference)
    - REGION/REGIONX parameter on JCL
    - IEFUSI or IEALIMIT exit
- Excessive use of LSQA/high private can affect user region
  - No capability to limit LSQA/high private – it's authorized!

For information about SMFLIMxx, see the following:

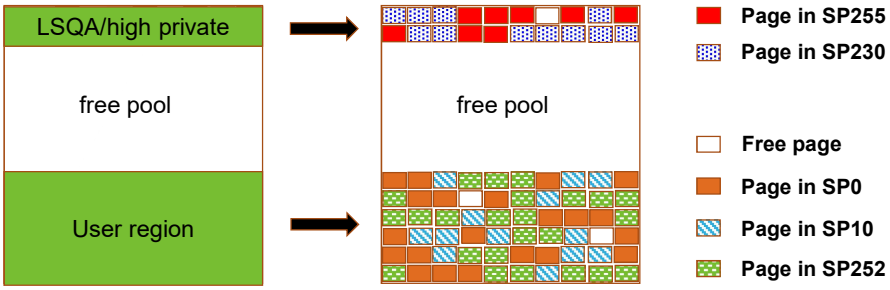
https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.3.0/com.ibm.zos.v2r3.ieae200/smflimxx.htm

Introducing subpools

SHARE
EDUCATE • NETWORK • INFLUENCE

- These LSQA, high private, and user region storage areas are comprised of pages of storage, each of which belongs to a **subpool**

A subpool is a collection of in-use pages of storage with related attributes



In the picture, we can see different pages belonging to different subpools intermingled in storage. However, pages from user region subpools such as SP0, SP10, and SP252 may not be interspersed with pages from LSQA subpools such as SP255, or with pages from high private subpools such as SP230. Note that LSQA SP255 pages may intermingle with high private SP230 pages since both are authorized storage subpools.

Note that there may be free pages interspersed within either unauthorized or authorized storage.

Subpool:
A collection of in-use pages with related attributes

SHARE
EDUCATE • NETWORK • INFLUENCE

- z/OS manages storage through the use of a variety of subpools
  - Different subpools, numbered 0 thru 255, support different storage needs
  - Subpool numbers and associated attributes are defined by the operating system
  - Attributes include:
    - LSQA, high private, or user region (as well as CSA and SQA)
    - Pageable versus fixed
    - Fetch-protected or not fetch-protected

See MVS Diagnosis: Reference Chapter 8 for subpool numbers and attributes

- Programs requesting storage provide an appropriate subpool number to VSM (plus a length) based on the program's storage requirements

There is a wide range of subpool numbers, but many of these subpools share the same attributes. The subpool number specified on a storage request will determine whether the storage being obtained is to be fixed or pageable; fetch-protected or not; LSQA, hi private, or user region (or SQA or CSA). There is also one other storage category called SWA that we don't discuss here, and there is a storage subtype called DREF which is actually specialized LSQA/SQA.

While most subpools support requests for storage above or below the line, there are a few subpools that are above the line only.

Assigning storage to subpools

SHARE
EDUCATE • NETWORK • INFLUENCE

- Pages of available local storage reside in an address space's free pool
- Pages of local storage are remapped from the free pool to a "subpool" as needed
  - Storage in a free pool is said to be "unallocated"
  - Storage assigned to a subpool is said to be "allocated"
- Pages of subpool storage are returned to the free pool when no longer needed

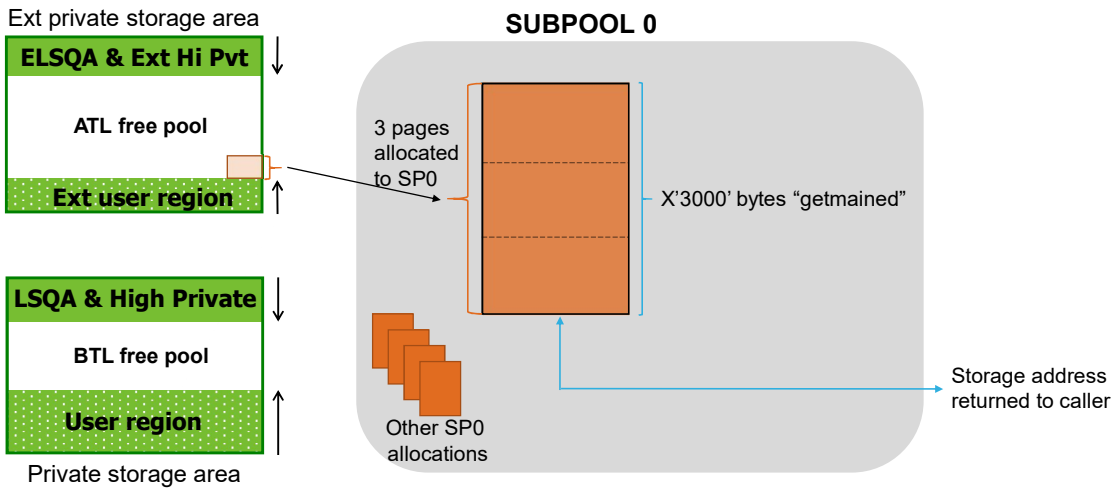
Key Concept: A subpool **grows** in size as storage requests for that subpool result in pages being assigned to it.

There is typically a large contiguous block of free pages between the top of the user region (unauthorized) storage and the bottom of authorized storage, there are also likely to be free pages scattered throughout these areas of storage. Any of these pages may be used to honor a storage request, bearing in mind the restriction that authorized and unauthorized storage may not intermingle.

An example

- A program requests **x'3000' bytes of above the line Subpool 0 (SP0)** storage
- Three contiguous pages are taken out of the low end of the above the line private storage free pool in that program's address space
- VSM assigns the 3 pages to SP0
- The address of the beginning of the X'3000'-byte area is returned to the program

Local storage growth



Because the request is for storage from a user region subpool, pages from the low end of the free page pool are used for the request. These are assigned to SP0, and the address of this 3-page range is returned to the caller.

Storage keys

- Every page of allocated storage has an associated key
  - Offers protection for the page
- Storage keys range from 0-15
  - System keys: 0-7
  - User keys: 8-15
- Storage key is determined based on the requested subpool**
  - LSQA is always KEY0
  - High private storage key is specified by requester
  - User region storage key is a function of the executing task's key

** See MVS Diagnosis: Reference Chapter 8 for more information on how the key is determined

Storage keys offer some degree of protection for a page. A program executing in a particular key can only update storage that has the same associated key.

The major exception is a program executing in key0. Key0 is the key of ultimate authority. A program executing in key0 is allowed to update nearly any storage on the system. (Exceptions: page protected storage, read-only storage, and the first X'200' bytes of each page of the PSAs.)

If storage is fetch-protected, then it can only be read by a program executing in the same storage key or by a program executing in key0.

Key definitions

SHARE
EDUCATE • NETWORK • INFLUENCE

- Key0 thru key7 are **system** keys (only usable by authorized code)
  - **Key0 – Used by operating system, has super authority**
  - Key1 – JES, APPC, TSO
  - Key5 – Data Management (DFSMS)
  - Key6 – CommServer
  - Key7 – IMS, DB2
- Key8 thru key15 are **non-system** keys
  - Key8 – Key most commonly used by unauthorized applications
  - Key9 – CICS users (transactions)

See Chapter 8 of the **MVS Diagnosis: Reference** manual for further information on common users of particular storage keys.

"Authorization" is a state of privilege on the operating system. Code is authorized if any one of the following is true:

1) The code is APF Authorized, that is, it is running under a program that resides in an authorized library and that program was indicated as being authorized when the link edit was performed.

2) The code is running in Supervisor state.

3) The code is running in Key0.

Being in Supervisor state carries more privilege than just being authorized.

Being in Supervisor state key0 carries even more privilege.

A program that is APF authorized has the ability to run in Supervisor state and/or in an execution key of 0.

Characteristics of types of local storage

SHARE
EDUCATE • NETWORK • INFLUENCE

- **LSQA**
  - 255 (mainly)
  - Authorized
  - Fixed, key0 storage
  - Address space-related
  - Holds system control blocks

- **High Private** subpools
  - 229, 230, 249
  - Authorized
  - Specifiable key
  - TCB-related
  - Special subpools for authorized application storage needs

- **User Region** subpools
  - 0 thru 132, 250 thru 252
  - TCB-related
  - Key not specifiable
  - Unauthorized
  - General purpose subpools for application storage needs

See MVS Diagnosis: Reference chapter 8 for additional subpool information.

LSQA is always key0 and always fixed. It is used by critical operating system code.

High private is used heavily by authorized applications and the subpool key is specifiable.

Note that high private storage is task-related but LSQA is not. Storage that is task-related will be automatically freed by the operating system when the task terminates. LSQA must always be explicitly freed.

User region is like high private's poor cousin. It too is task-related and it is used by applications running in the address space. However, it is unauthorized and as such is relegated to using a generic storage key associated with the job, rather than being able to determine its own storage key.

VSM uses the same types of control blocks to describe user region as it uses for high private.

Subpool management rules

- Storage is allocated to a subpool in one-page (4K) multiples
- Storage with different attributes cannot coexist on same page
  - Storage belonging to different **subpools** cannot occupy the same page
  - Storage with different storage **keys** cannot occupy the same page
  - Storage belonging to different **TCBs** cannot occupy the same page
- Authorized storage (LSQA, high private) cannot intermix with unauthorized storage (user region)
- If there is insufficient free storage above the line to fulfill an above the line request, VSM will look for storage below the line**
- For storage requests < 1 page, VSM generally gives out storage at the high end of a page first

** Exception: Some subpools are defined as above the line only subpools.

VSM maintains segregation at a page level. Storage on the same page will have the same subpool, same key, and as applicable, the same owning TCB.

As with everything else in VSM, there are some minor exceptions to this rule.

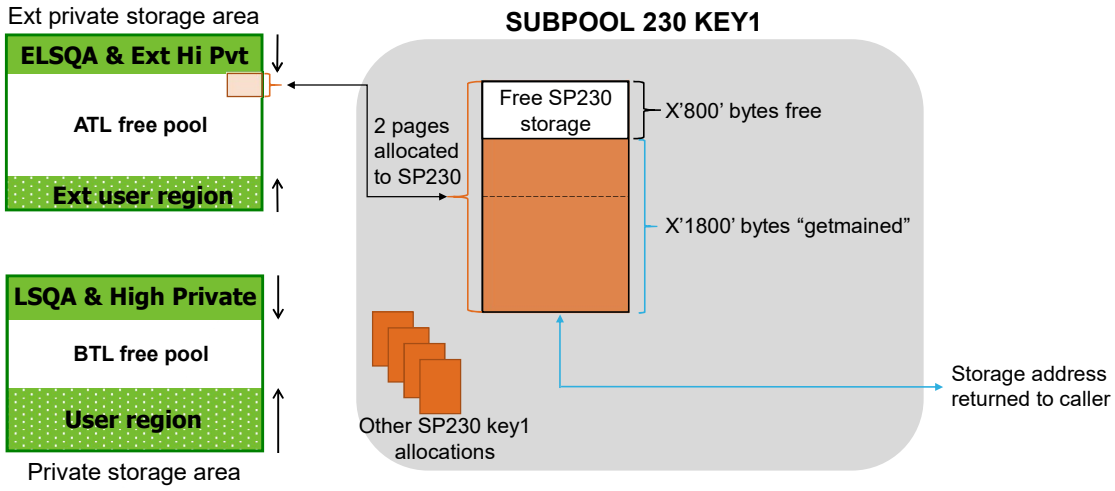
Some subpools are defined as below the line only subpools. Some subpools are defined as above the line only subpools. If there is insufficient free storage above the line to honor a request for storage in an above the line only subpool, the system will abend rather than trying to obtain storage from below the line.

Another example

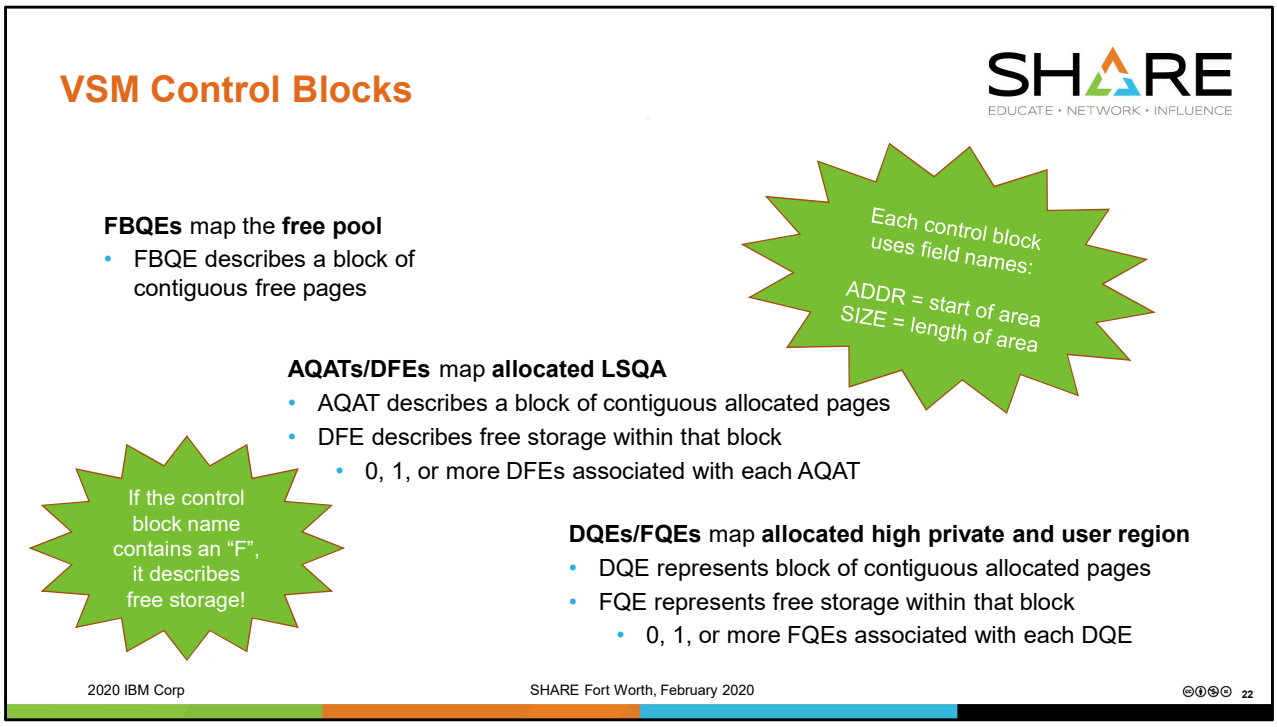
- A program requests **x'1800' bytes of above the line SP230 KEY1** storage
- Two contiguous pages are taken out of the high end of the above the line private storage free pool in that program's address space
- VSM assigns the 2 pages to SP230 key1
- VSM makes note that x'800' bytes of the 2 pages assigned to SP230 key1 are still free
- The address of the beginning of the X'1800'-byte area is returned to the program

In this example, 2 pages of storage are allocated (assigned) to SP230 key1. However only x'1800' bytes of the x'2000' bytes in the 2-page allocation are actually getmained. The remaining x'800' bytes are free, but allocated to SP230 key1, which means they can only be used to satisfy additional requests for SP230 key1 under the same TCB. These x'800' bytes cannot be used to satisfy a getmain for any other subpool or key or TCB.

Local storage growth



READING THE VSMDATA REPORT



In the previous example where x'1800' bytes was getmained from SP230 key1 (high private), VSM would locate an FBQE with a SIZE field of x'2000' or greater. It would map 2 pages to a subpool by reducing the FBQE SIZE field size by x'2000' and building a DQE with SIZE of x'2000'. Then it would build an FQE associated with the DQE which had a SIZE field of x'800'.

If the FBQE contained an ADDR field of x'8123000' and a SIZE field of x'5000', after the getmain, we would see control blocks with the following content:

FBQE: ADDR 00813000 SIZE 3000

DQE: ADDR 00816000 SIZE 2000

FQE: ADDR 00817800 SIZE 800

Viewing VSM control blocks

SHARE
EDUCATE • NETWORK • INFLUENCE

- To format VSM's local storage control blocks in a dump:
  - **IPCS VERBX VSMDATA 'NOGLOBAL SUM JOBNAME(jjjjjjjj)'**
  - **IPCS VERBX VSMDATA 'NOG SUM ASID(n)'**
    where n is decimal ASID
  - **IPCS VERBX VSMDATA 'NOG SUM'**
    if only one address space in dump**

** Console dump may default to ASID 1

VSM tends to think of things in “opposites”. It represents storage that is free rather than storage that is getmained. Instead of telling the VSM formatter “show me local storage,” we tell it “don't show me global.”

Omitting the SUM parameter gives a much more detailed and much less readable report. This report is occasionally used within VSM L2 when debugging VSM-internal issues, but has no relevance for out of storage analysis.

Example of FBQEs (free pool)

SHARE
EDUCATE • NETWORK • INFLUENCE

```
 --FREE BLOCKS OF STORAGE

     FBQE: Addr 0003A000 Size     8000      TCB:    n/a   SP/K: n/a
     FBQE: Addr 00049000 Size    18000      TCB:    n/a   SP/K: n/a
     FBQE: Addr 00062000 Size   A40000      TCB:    n/a   SP/K: n/a
     FBQE: Addr 00AA3000 Size     2000      TCB:    n/a   SP/K: n/a
     FBQE: Addr 00AA8000 Size     1000      TCB:    n/a   SP/K: n/a

Extended Address Space Region Descriptor data follows

   --FREE BLOCKS OF STORAGE

     FBQE: Addr 22B00000 Size     1000      TCB:    n/a   SP/K: n/a
     FBQE: Addr 22B02000 Size    30000      TCB:    n/a   SP/K: n/a
     FBQE: Addr 22B33000 Size    26000      TCB:    n/a   SP/K: n/a
     FBQE: Addr 22B84000 Size    7C000      TCB:    n/a   SP/K: n/a
     FBQE: Addr 22C01000 Size 5C7F4000      TCB:    n/a   SP/K: n/a
```

There is an 8-page block of free storage at address 3A000

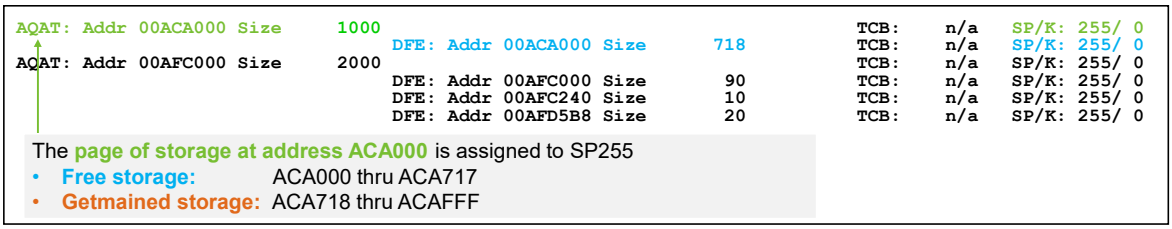
FBQE Addr
- Defines the **beginning of a free block** of storage
- Always **page-bounded**

FBQE Size
- Defines **length of the free block** of storage
- Always a **multiple of a page** (X'1000')

Here we see FBQEs representing below the line free areas (in this case, for private storage), formatted in ADDR order, followed by FBQEs representing above the line free areas, formatted in ADDR order.

Free pool storage is always managed in page multiples.

Example of AQATs/DFEs (LSQA)



The page of storage at address ACA000 is assigned to SP255
- Free storage: ACA000 thru ACA717
- Getmained storage: ACA718 thru ACAFFF

AQAT Addr
- Defines the start of an allocated block of storage
- Always page-bounded

AQAT Size
- Defines length of the allocated block of storage
- Always a multiple of a page (X'1000')

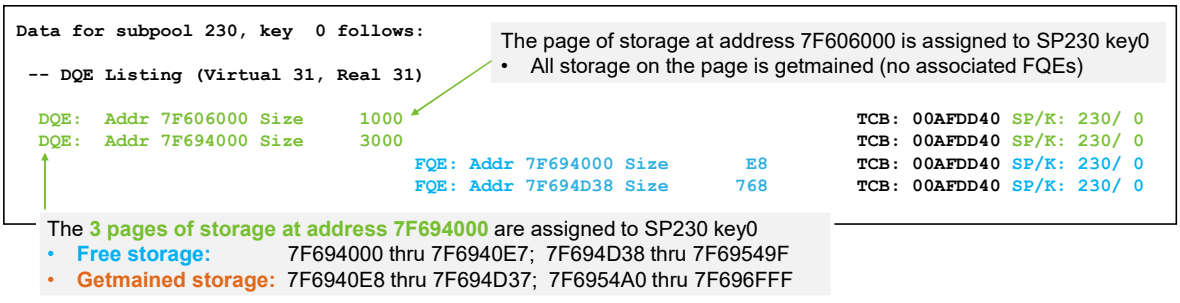
DFE Addr
- Defines the start of a free area of storage within an allocation

DFE Size
- Defines length of a free area of storage within an allocation

An AQAT represents pages of allocated LSQA. Their corresponding DFEs describing free storage within the allocation (if any) are formatted underneath and to the right.

The GETMAINed storage is the inverse of the storage that is free within the allocation.

Example of DQEs/FQEs (high pvt; user region)



DQE Addr
- Defines the **start of an allocated block** of storage
- Always **page-bounded**

DQE Size
- Defines **length of the allocated block** of storage
- Always a **multiple of a page** (X'1000')

FQE Addr
- Defines the **start of a free area** of storage within an allocation

FQE Size
- Defines **length of a free area** of storage within an allocation

FQEs are to DQEs as DFEs are to AQATs. DQEs and AQATs both represent allocated pages of storage, that is, pages of storage that have been subpool-assigned. They can represent a single page or a block of contiguous pages. FQEs and DFEs represent free storage within the allocation. There may be 0, 1, or more FQEs associated with a DQE, and there may be 0, 1, or more DFEs associated with an AQAT.

The term “block” is used extensively in this presentation and refers to 1 or more contiguous pages of storage.

Note that not only do we have subpool and key identified to the right of these DQE and FQE lines, but we also have a TCB specified. This is because SP230 (high private) is task-related.

Layout of VSMDATA local storage report

SHARE
EDUCATE • NETWORK • INFLUENCE

IPCS VERBX VSMDATA 'NOGLOBAL SUM JOBNAME(jjjjjjj)'

- LSQA subpools in order by subpool number
- FBQEs in ADDR order
- For each TCB
  - SWA subpools in subpool/key order
  - High private subpools in subpool/key order
  - User region subpools in subpool/key order
- Local storage map
- Summary of key fields from VSM Local Data Area (LDA)
- Summary of subpool totals, each also broken down into BTL / ATL

Layout of VSMDATA local storage report (cont)

SHARE
EDUCATE • NETWORK • INFLUENCE

For each subpool:

- Headers identifying the following VSM categories
  - Virtual 24, Real 24
  - Virtual 24, Real 31
  - Virtual 24, Real 64
  - Virtual 31, Real 31
  - Virtual 31, Real 64
  - Virtual 31, Real PAGEFRAMESIZE1MB
- AQAT/DFE or DQE/FQE control blocks in ADDR order under appropriate header
- Total allocated storage for subpool/key, with BTL / ATL breakdown
  - Except for LSQA, which displays totals by real storage usage (real 24 total, real 31 total, real 64 total)

If a subpool does not have any allocations in a particular category, then that section header is not formatted in the report.

Example of LSQA layout

```
Data for LSQA subpool 255 follows:

   --Virtual 24, Real 31 AQAT/DFE listing (Address order)
     AQAT: Addr 008D1000 Size     1000
                                       DFE: Addr 008D1000 Size      D58
     AQAT: Addr 008FC000 Size     1000

   --Virtual 31, Real 31 AQAT/DFE listing (Address order)
     AQAT: Addr 7FFD1000 Size     1000
                                       DFE: Addr 7FFD1000 Size      130

   ***** Subpool 255 (Real 31) Allocation:  3000 ( 2000 Below, 1000 Above )

--Virtual 24, Real 31 DFEs for subpool 255 (Size order)
   ...
--Virtual 31, Real 31 DFEs for subpool 255 (Size order)
   ...

   --Virtual 24, Real 64 AQAT/DFE listing (Address order)
     AQAT: Addr 008E6000 Size     1000
                                       DFE: Addr 008E6000 Size      8C0
     AQAT: Addr 008FD000 Size     3000

   ***** Subpool 255 (Real 64) Allocation:  4000 ( 4000 Below, 0 Above )
```


There is a bit of clutter in the control block displays for LSQA. Also, while there are lines providing subtotals for the different real storage categories in the subpool, there is no final total for that subpool in this section of the report. (There is a total in the subpool summary.)

When examining AQAT/DFE structures in a storage analysis, it is important to be aware that you may need to examine multiple sections in the subpool. Note that while there are numerous possible sections, only those with actual entries will appear in the report. For example, if the subpool has no storage in the “Virtual 24, Real 24” category, then that section will not be listed at all.

The real storage information has no bearing on VSM storage analysis.

Example of task-related storage layout

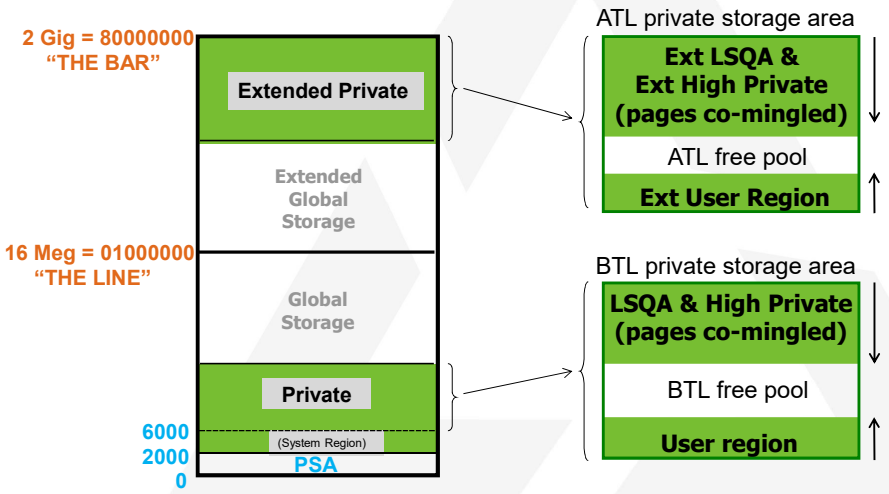
```
Data for TCB at address 008FE050
   Data for subpool 229, key  0 follows:
    -- DQE Listing (Virtual 31, Real 64)
     DQE:  Addr 7FFED000 Size     1000
                                        FQE: Addr 7FFED000 Size      F28
   ***** Subpool  229, key  0 Total alloc:  1000 ( 0 Below, 1000 Above )
   Data for subpool 229, key  1 follows:
    -- DQE Listing (Virtual 31, Real 31)
     DQE:  Addr 7F6C9000 Size     C000
     DQE:  Addr 7F72B000 Size    18000
     DQE:  Addr 7F74A000 Size     1000
     DQE:  Addr 7FFB2000 Size     D000
     DQE:  Addr 7FFC4000 Size     1000
    -- DQE Listing (Virtual 31, Real 64)
     DQE:  Addr 7F74F000 Size     1000
                                        FQE: Addr 7F74F000 Size      960
   ***** Subpool  229, key  1 Total alloc:  34000 ( 0 Below, 34000 Above )
```


On a TCB by TCB basis, each subpool/key for which the TCB owns storage is formatted in the report. DQEs/FQEs are listed in ADDR order within the proper real storage category for the subpool.

Note that it is possible for a TCB to have multiple of the same subpool number but with different keys.

VSM PROBLEM DIAGNOSIS

Local storage management (reminder)



Authorized storage is given out from the high end of the private storage areas

Unauthorized storage is given out from the low end of the private storage areas

Out of local storage conditions

- When VSM cannot fulfill a request for local storage, it issues **ABEND878** or **ABEND80A** with appropriate reason code:
  - **RC x'0C'** A request for LSQA failed
  - **RC x'10'** A request for user region, high private, or SWA failed
- Consider our picture
  - An un-limited user region can grow upwards and squeeze out LSQA/high private
  - LSQA or high private can grow down and squeeze out user region
  - LSQA can squeeze out high private; high private can squeeze out LSQA

Conclusion: The victim is not necessarily the "bad guy"!!

Steps in diagnosing ABEND878 or ABEND80A

SHARE
EDUCATE • NETWORK • INFLUENCE

1. Determine failing GETMAIN size/subpool**
2. Review local storage map and FBQEs to understand storage usage
3. Look for subpools that may be too large
4. Examine suspicious subpools for patterns
5. If pattern found, ID it by browsing storage and searching data base
6. If pattern cannot be ID'd, G/F trace
7. If no subpool/pattern found, check recently obtained storage
8. Absence of a pattern suggests tuning

**This step is usually not critical to successful problem resolution

In most storage errors, the victim who suffered the ABEND878 or ABEND80A is NOT the "bad guy". Getting information about the victim is not particularly helpful. Steps 2 thru 4 in our analysis are the more important ones, allowing us to first assess which area of private storage appears to be the problem (region or LSQA/high private? below or above the line?), then drill down to suspicious subpools, and then to drill further into the VSM control blocks defining that subpool, looking for patterns of abuse.

Note that, because step #1 is usually not critical, this implies that you do not need an actual abend dump to debug a storage growth condition. Often a console dump can be taken of a storage creep BEFORE abends begin to occur, and a diagnosis made by applying steps 2 thru 8.

Step #1 *does* become important when dealing with an abend where the abend is due to code requesting storage of an incorrect and unreasonably large length. In this case it is helpful to determine the length to verify that it was excessive. It is, of course, also important to identify the issuer of the bad storage request. Do not use the error PSW. This will point into VSM since it detected the abend condition. To identify who made the storage request, check the entries in the system trace table just prior to the ABEND878 or ABEND80A. It may also be necessary to check Register 14 from the registers at time of error.

ABEND878 error information

Do: **IP ST REGS** against dump of ABEND878

```
PSW=070C1000 815B3B50
    (Running in PRIMARY, key 0, AMODE 31, DAT ON)
    DISABLED FOR PER
   ASID(X'0019') 015B3B50. IEANUC01.IGVVSERR+0E40 IN READ ONLY NUCLEUS

  General purpose register values
     0-1  00000010_84000000  00000000_84878000 <-- abend code
     2-3  FFFFFFFF_00000602  FFFFFFFF_00000272 <-- subpool number**
     4-5  FFFFFFFF_008E6968  FFFFFFFF_008E68E0
     6-7  FFFFFFFF_815AEC68  FFFFFFFF_00F57000
     8-9  FFFFFFFF_00000000  FFFFFFFF_000EF888 <-- Length**
    10-11 FFFFFFFF_00000000  FFFFFFFF_7FFFDDC0
    12-13 FFFFFFFF_00000081  00000000_00006F60
    14-15 00000000_00007F3E  00000000_00000010 <-- reason code
```

** If Reg3 points to an ASCB control block, then VSM was entered via PC 30B STORAGE OBTAIN. Get subpool number and length from the linkage stack entry. See speaker notes for more information.

How do you recognize an ASCB address? An ASCB address is a below the line SQA address that ends in x'80' or x'00'. If you browse or list an ASCB address, you will see an 'ASCB' eyecatcher in the first word.

For STORAGE OBTAIN (PC 30B), Register 0 on entry contains the length of storage being requested, and byte 2 (counting from 0) of Register 15 contains the subpool number.

Information about subpool and length requested can also be found in a system trace entry. For the abending unit of work locate the trace entry immediately before the SVC D for the ABEND878:

- For an SSRV 132 STORAGE OBTAIN entry, an SSRV 78 GETMAIN entry, or an SVC 78 GETMAIN entry:
  - Byte 2 of the Unique-1 field indicates the subpool number
  - The Unique-2 field indicates the length of storage requested

ABEND80A error information

SHARE
EDUCATE • NETWORK • INFLUENCE

Do: **IP ST REGS** against dump of ABEND80A

```
PSW=070C1000 815B3B50
    (Running in PRIMARY, key 0, AMODE 31, DAT ON)
    DISABLED FOR PER
   ASID(X'0019') 015B3B50. IEANUC01.IGVVSERR+0E40 IN READ ONLY NUCLEUS

  General purpose register values
     0-1  00000010_84000000  00000000_8480A000
     2-3  FFFFFFFF_00000602  FFFFFFFF_00000272
     4-5  FFFFFFFF_008E6968  FFFFFFFF_008E68E0
     6-7  FFFFFFFF_815AEC68  FFFFFFFF_00F57000
     8-9  FFFFFFFF_00000000  FFFFFFFF_FA006000
    10-11 FFFFFFFF_00000000  FFFFFFFF_7FFFDDC0
    12-13 FFFFFFFF_00000081  00000000_00006F60
    14-15 00000000_00007F3E  00000000_00000010
```

Abend Code

Subpool number in first byte; length in last 3 bytes

Return Code

Information about subpool and length requested can also be found in a system trace entry. For the abending unit of work locate the trace entry immediately before the SVC D for the ABEND80A:

- For an SSRV A GETMAIN or an SVC A GETMAIN entry:
  - Byte 0 of the Unique-2 field indicates the subpool number
  - Bytes 1 thru 3 of the Unique-2 field indicates the length of storage requested

Let's go to the map!!

SHARE
EDUCATE • NETWORK • INFLUENCE

```
    LOCAL STORAGE MAP
 _____________________________
|                             |80000000  <- Top of Ext. Private
| Extended                    |
| LSQA/SWA/229/230            |
|_____________________________|7EF24000  <- ELSQA Bottom
|                             |
| (Free Extended Storage)     |1E7A8000  <- Max Ext. User Region Address***
|_____________________________|1E7A8000  <- Ext. User Region Top
|                             |
| Extended User Region        |
|_____________________________|18600000  <- Ext. User Region Start
:                             :
: Extended Global Storage     :
=========================================<- 16M Line
: Global Storage              :
:_____________________________:  A00000  <- Top of Private
|                             |
| LSQA/SWA/229/230            |  A00000  <- Max User Region Address
|_____________________________|  96B000  <- User Region Top**
|                             |
|                             |
| User Region                 |
|_____________________________|    6000  <- User Region Start
: System Storage              :
:_____________________________:       0

**   User Region Top has hit bottom of LSQA
***  Ext. User Reg. Top has hit Max. Ext. User Reg. Address
```

IP VERBX VSMDATA 'NOG SUM ASID(nn)'

- Job is limiting user region growth ATL (Max Ext. User Region Address does not equal Top of Ext. Private)
- Job is not limiting user region growth BTL
- **VSMDATA has flagged warning conditions**

Conclusion: User Region problem

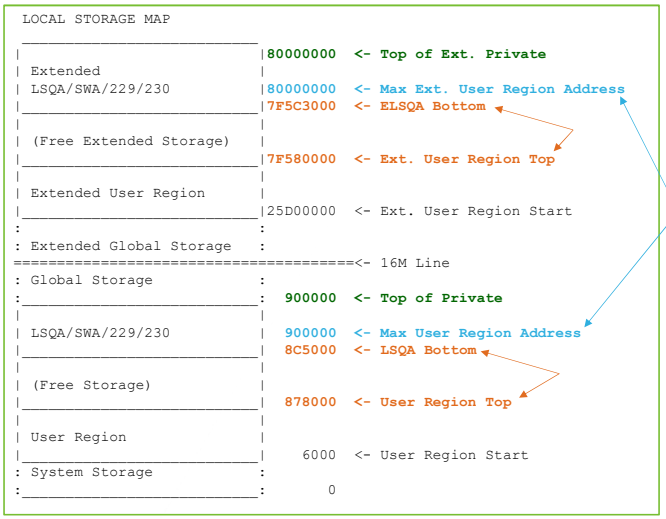
In this example of a local storage map formatted in the VSMDATA report, we can see that the upward growth of user region and the downward growth of LSQA/high private are documented. The report identifies the User Region Top and the LSQA Bottom, both above and below the line. However, in this example, you will see that the BTL User Region Top has hit the bottom of LSQA, thus a single line marks this on the map, the line is flagged with an asterisk, and a corresponding comment is added below the map.

Similarly, the map identifies the maximum address that user region below the line is allowed to grow up to, and the maximum address that user region above the line is allowed to grow up to. In this example, the above the line user region has hit its max, so once again only one line is displayed. This line is marked with an asterisk, and further clarification is added below the map.

The flagged lines and comments below the map are very helpful in understanding the storage picture. In this case we have a job where user region growth was limited above the line but not below. User region above the line grew until it hit its limit, and then started consuming storage below the line. Eventually the below the line user region grew up so far that it hit the bottom of below the line LSQA/high private.

Let's go to another map!!

SHARE
EDUCATE • NETWORK • INFLUENCE



IP VERBX VSMDATA 'NOG SUM ASID(nn)'

- Note that **user region top** is very close to **LSQA bottom** both BTL and ATL
- **Job is not limiting user region BTL or ATL**
- LSQA areas look normal in size
  - LSQA size usually <1Meg
  - ELSQA size usually <X'40'Meg

Conclusion: User Region problem

In this example, which we will explore further as we go through the VSMDATA report, we see that there is no limit on user region growth either below or above the line. Rather, the max user region is equal to the address of the top of the local storage area.

While we don't have any lines noting that user region has hit the bottom of LSQA/high private either below or above the line, we can see that the two areas are close to bumping both below and above the line. A small storage request could be accommodated either place, but a request for X'50000' bytes (for example) would fail.

Related key fields

SHARE
EDUCATE • NETWORK • INFLUENCE

```
Summary of Key Information from LDA (Local Data Area) :

STRTA  =     6000  (ADDRESS of start of private storage area)
SIZA   =   8FA000  (SIZE of private storage area)
CRGTP  =   878000  (ADDRESS of current top of user region)
LIMIT  =   8FA000  (Maximum SIZE of user region)
LOAL   =   872000  (TOTAL bytes allocated to user region)
HIAL   =    3B000  (TOTAL bytes allocated to LSQA/SWA/229/230 region)
SMFL   = FFFFFFFF  (IEFUSI specification of LIMIT)
SMFR   = FFFFFFFF  (IEFUSI specification of VVRG)
```

Note: ATL equivalent fields are also reported

Total storage allocated in BTL User Region

Total storage allocated in BTL LSQA/high private

- From the storage map, you can easily calculate the **difference** between:
  - **Bottom and top of user region --- compare to LOAL**
  - **Bottom and top of LSQA/high private --- compare to HIAL**
- A large disparity would suggest fragmentation. Check FBQEs for more info!

No disparity in our example – proceed to looking at subpool totals

Most of the data in the key fields report is represented pictorially in the map. However, a pair of fields that can offer additional insight is the LDALOAL field and the LDAHIAL field. Each of these has an above the line equivalent as well: LDAELOAL and LDAEHIAL. As an example, if below the line user region looks very large in the map but LDALOAL is relatively small, this would suggest that there are a lot of “holes” or free blocks within the user region. This is known as fragmentation. The next step when fragmentation is suspected is to consult the FBQEs to figure out how the free blocks of storage are distributed. Fragmentation problems can be tricky to debug because you are dealing with the question of what storage **isn't** there rather than what storage **is** there.

Luckily, our data above does not suggest a fragmentation problem. Rather, the storage is very compactly or densely allocated. We can proceed comfortably with our assumption that we have a problem with a user region subpool becoming overgrown. Now we need to figure out which one it is.

Examining subpool totals

SHARE
EDUCATE • NETWORK • INFLUENCE

```
LOCAL SUBPOOL USAGE SUMMARY

     TCB/OWNER    SP#   KEY      BELOW     ABOVE     TOTAL
     ---------    ---   ---      -----     -----     -----
       8E6E88       0     8       1000         0      1000
       8E6968       2     8     870000  59880000  5A0F0000
        LSQA      205     0          0     39000     39000
        LSQA      215     0          0     D000      D000
        LSQA      225     0          0     13000     13000
       8FE050     229     0          0     1000      1000
       8FD0D0     229     0          0     91000     91000
       8FF890     229     0          0     2000      2000
       8FE050     229     1          0     34000     34000
         . . . .
       8FE050     230     8       1000     2000      3000
       8FE050     230     9       1000     2000      3000
       8FF890     236     1       D000     18000     25000
       8FF890     237     1       6000     1A000     20000
       8E6E88     237     1       F000     18000     27000
       8FE050     249     1          0     2000      2000
       8E6968     251     8       1000        0      1000
        LSQA      255     0       6000     B7000     BD000
```

We think we have a user region problem

- User region subpools: 0-132, 250-252
- Do any user region subpools look overgrown?
  - YES - Check subpool detail (DQE's/FQE's) in the upper section of the VSMDATA report
  - NO - There is an alternative technique that we will mention later

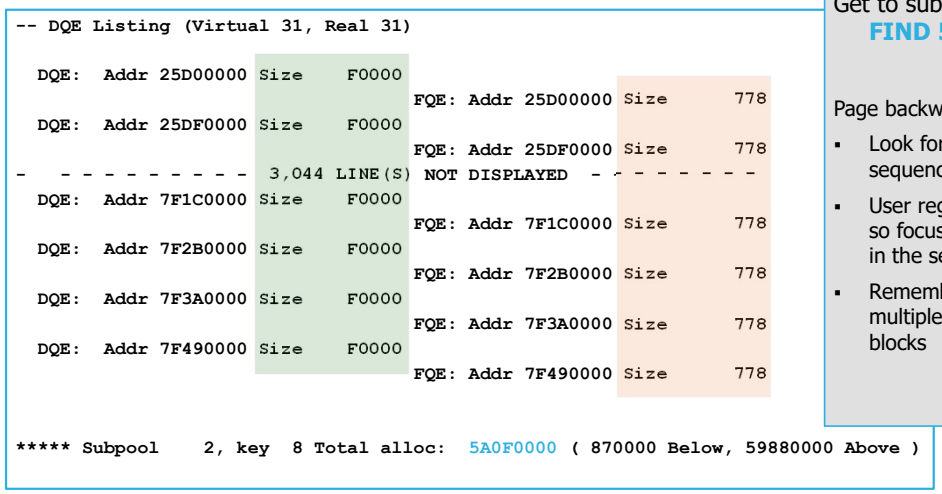
Often it really is pretty easy to identify the problem subpool. Other times you have to work a little harder, checking a few possible candidates.

The subpool summary is formatted out in ascending subpool order, and by increasing keys within subpools. If multiple TCBs have the same subpool/key combination, these are shows on consecutive lines.

The report breaks the subpools down into below the line total, above the line total, and grand total for each subpool/key/TCB combination.

Once a candidate as a problem subpool is identified, look for its AQAT/DFE or DQE/FQE detail in the upper portion of the VSMDATA report. For non-LSQA subpools, its total or a subtotal of storage usually provides a fairly unique search argument. This won't work for cases where an LSQA subpool is suspect because the grand totals for the subpool don't appear in the upper portion of the report. Instead do “FIND LSQA” from the top, and repeat until you come to the subpool you are interested in.

Examining user region subpool control blocks



Get to subpool Total line:
FIND 5A0F0000 FIRST

Page backward (PF7)

- Look for DQE/FQE pattern sequence
- User region storage grows up so focus on higher addresses in the section
- Remember there may be multiple sections of control blocks

When checking the DQEs/FQEs of user region subpools, remember that this storage grows from lower addresses to higher addresses. Since you want to look at the control blocks for the most recently allocated storage, you will want to scroll down to the highest addresses in the section. Again remember that there may be multiple sections that need to be checked for the subpool of interest.

You know you've found the bad guy when you see a pattern like this. Someone has been requesting storage of length X'EF888' (F0000-778) over and over again. Each time, VSM reassigns X'F0' pages from the free pool to SP2, then carves out the X'778' byte chunk to make the amount of storage left over be the requested amount of X'EF888' bytes. Such a pattern is quite typical in local storage growth problems.

Reminder:
Steps in diagnosing ABEND878 or ABEND80A

SHARE
EDUCATE • NETWORK • INFLUENCE

1. Determine failing GETMAIN size/subpool**
2. Review local storage map and FBQEs to understand storage usage
3. Look for subpools that may be too large
4. Examine suspicious subpools for patterns
5. If pattern found, ID it by browsing storage and searching data base
6. If pattern cannot be ID'd, G/F trace
7. If no subpool/pattern found, check recently obtained storage
8. Absence of a pattern suggests tuning

**This step is usually not critical to successful problem resolution

So we've found a pattern, and now the trick is to look for eyecatchers or pointers in storage that will allow us to identify who is responsible for it. You may certainly try looking for a GETMAIN for the identified storage within the system trace, but most often it is not there because the GETMAINs are occurring to few and far between.

When searching the IBM defect data base for possible APARs, represent the subpool as Spnnn, the key as Keyn, and include any relevant eyecatchers.

How do steps differ for LSQA/high private?

SHARE
EDUCATE • NETWORK • INFLUENCE

- Overgrown subpool may be **more subtle**
  - LSQA and high private storage generally comprise a smaller area of local storage
- **No subpool total line for LSQA** in the control block section
  - Find the right subpool section by doing: **FIND LSQA**
- **LSQA uses AQATs/DFEs** rather than DQEs/FQEs
- LSQA and high private storage **grow down**, so a **pattern would more likely be found in the lower addresses within a section**

If your storage map analysis showed that LSQA/high private appeared overgrown either above or below the line (criteria were provided on the slide that showed the map), then you must look in the subpool summary for overgrown high private or LSQA subpools. These may not be as dramatically overgrown as the user regions subpools tend to be. You may have to try a few different candidates, checking the DQE/FQE or AQAT/DFE control blocks of each, looking for patterns. Because of the way LSQA (nd SQA) are managed, patterns in AQATs/DFEs are a little looser, but still recognizable.

Remember that LSQA and high private storage grow from higher addresses to lower addresses, so when checking their DQEs/FQEs or AQATs/DFEs, look at the lower addresses rather than the higher addresses. After all, you want to be looking at the most recently allocated storage.

What if no suspicious subpool is found?

SHARE
EDUCATE • NETWORK • INFLUENCE

VSMDATA Report sorting technique *

* This sorting technique works for the VSMDATA global storage report as well.

- **IP VERBX VSMDATA 'NOG SUM ASID(nn)'**
- SORT 115 122
- FIND SP/K FIRST
- If desired, a block delete can be done to remove all preceding lines

The report is now sorted in ascending ADDRess order

- **User region problem** - Find the address of the top of the user region (per map) and look backwards for patterns
- **LSQA/high private problem** - Find the address of the bottom of the LSQA/high private and look forwards for patterns

When all else fails, identify where the most recently allocated storage lives (for local, this would be the bottom of LSQA/high private and the top of user region; for global storage, this is typically the bottom of CSA). Pinpoint this address range in the sorted VSMDATA report, then look forward and backward for patterns. Because storage may be a little more "mixed together" when looking at it this way, the pattern may be a little less obvious than our earlier example, but if one is there, you should be able to pick it out.

Storage creep, no abend (yet!)

SHARE
EDUCATE • NETWORK • INFLUENCE

- Get a console dump
- Use the same diagnostic procedure but skip step 1
- **Comparison dumps can prove helpful!**
  - Take 2 console dumps separated by enough time to show growth
  - Compare storage maps and subpool summaries across the 2 dumps

Analyzing a storage creep is not that much different from analyzing a storage abend. Skip step 1 of our diagnostic steps since there is no abend information to review. However, you can still consult the map looking at how big user region and LSQA/high private are, and you can still look for overgrown subpools using the subpool summary and the control block detail.

If storage is creeping up slowly, you may have the luxury of being able to get a second dump for comparison, taken long enough after the first dump such that growth will be evident. Do a comparison between the maps in the two dumps, and between the subpool summaries. An area of growth may be obvious. If so, go look at the control blocks for that subpool in the upper section of the report. See if you can spot a pattern.

Pattern found! – but can't ID it ☹ Now what??

SHARE
EDUCATE • NETWORK • INFLUENCE

GETMAIN / FREEMAIN/STORAGE trace

- Used when pattern is found but cannot be identified
- Requires problem recurrence
- Traces all forms of GETMAIN, FREEMAIN, and STORAGE requests
- Requires GTF trace*, started with USR(F65) option
- Activated through DIAGxx** parmlib member
  - Can filter by ASID, jobname, subpool, key, storage length
- When tracing SP240, SP250, SP0, or SP252, trace all four

* See MVS Diagnosis: Tools and Service Aids for more information
** See MVS: Initialization and Tuning Reference for more information

If you've found a pattern but don't know who did it in spite of looking for eyecatchers and chasing pointers within the storage, you may been to resort to running with the VSM GETMAIN/FREEMAIN/STORAGE trace and waiting for a recurrence. This trace uses GTF trace processing and provides parameters which allow you to tailor the output by subpool, key, jobname, ASID, and more. You can even trace by address range or getmain length.

SHARE
EDUCATE • NETWORK • INFLUENCE

Any questions?

Thank you for joining us!!