Session 16086
z/OS Virtual Storage Debugging:
How to solve 80A, 878, & related abends

MVS Core Technologies Project – August 7th, 2014

Patty Little
IBM Poughkeepsie
plittle@us.ibm.com

Trademarks

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®**

* Registered trademarks of IBM Corporation

The following are trademarks or registered trademarks of other companies.

- Java and all Java-related trademarks and logos are trademarks of Sun Microsystems, Inc., in the United States and other countries.
- Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.
- Microsoft, Windows and Windows NT are registered trademarks of Microsoft Corporation.
- UNIX is a registered trademark of The Open Group in the United States and other countries.
- SET and Secure Electronic Transaction are trademarks owned by SET Secure Electronic Transaction LLC.

* All other products may be trademarks or registered trademarks of their respective companies.

Notes:

Performance is in Internal Throughput Rate (ITR) ratio based on measurements and projections using standard IBM benchmarks in a controlled environment. The actual throughput that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput improvements equivalent to the performance ratios stated here.

IBM hardware products are manufactured from new parts, or new and serviceable used parts. Regardless, our warranty terms apply.

All customer examples cited or described in this presentation are presented as illustrations of the manner in which some customers have used IBM products and the results they may have achieved. Actual environmental costs and performance characteristics will vary depending on individual customer configurations and conditions.

This publication was produced in the United States. IBM may not offer the products, services or features discussed in this document in other countries, and the information may be subject to change without notice. Consult your local IBM business contact for information on the product or services available in your area.

All statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only.

Information about non-IBM products is obtained from the manufacturers of those products or their published announcements. IBM has not tested those products and cannot confirm the performance, compatibility, or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Prices subject to change without notice. Contact your IBM representative or Business Partner for the most current pricing in your geography.

Agenda

- What is VSM?
- Types of virtual storage
- VSM's storage management strategy
- Subpools and keys
- VSM control blocks
- IPCS VSMDATA report formats
- Out of storage abend information
- Debugging local storage shortages
- Debugging global storage shortages
- CSA tracker
- Common VSM problems

We will discuss the different storage managers, how they work, and what they manage.

What is VSM?

VSM – Virtual Storage Manager

- Manages virtual storage below the bar
  - (RSM manages virtual storage above the bar)
- Supports GETMAIN/FREEMAIN and STORAGE OBTAIN/RELEASE requests
- Issues ABEND878/ABEND80A to indicate "out of virtual storage" conditions

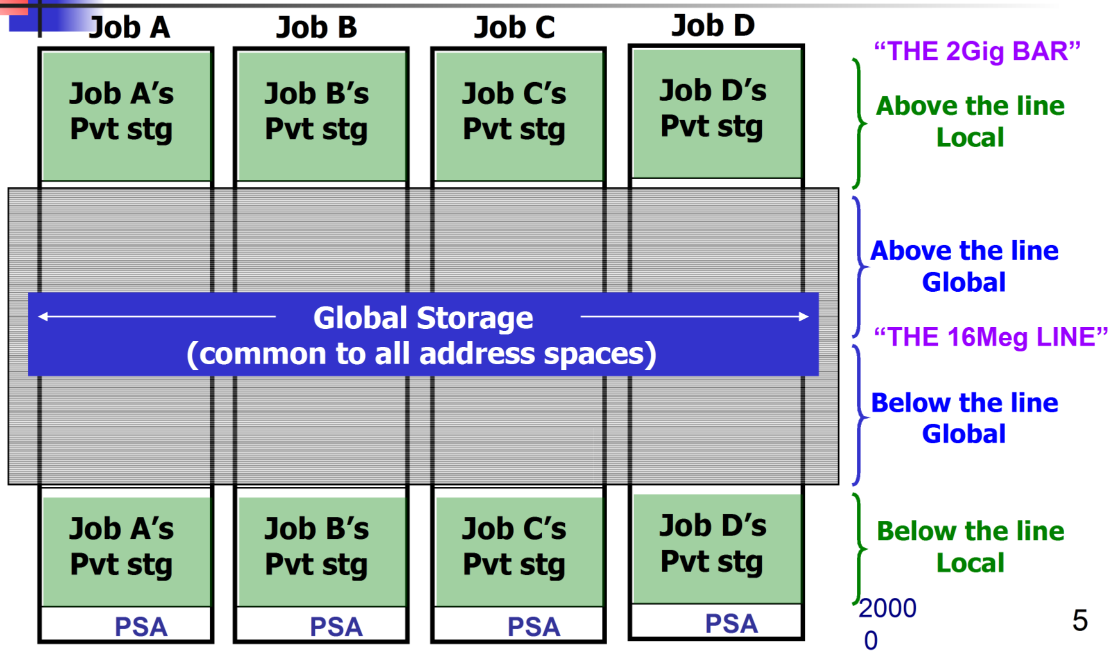
4

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.

Global vs Local Storage



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.

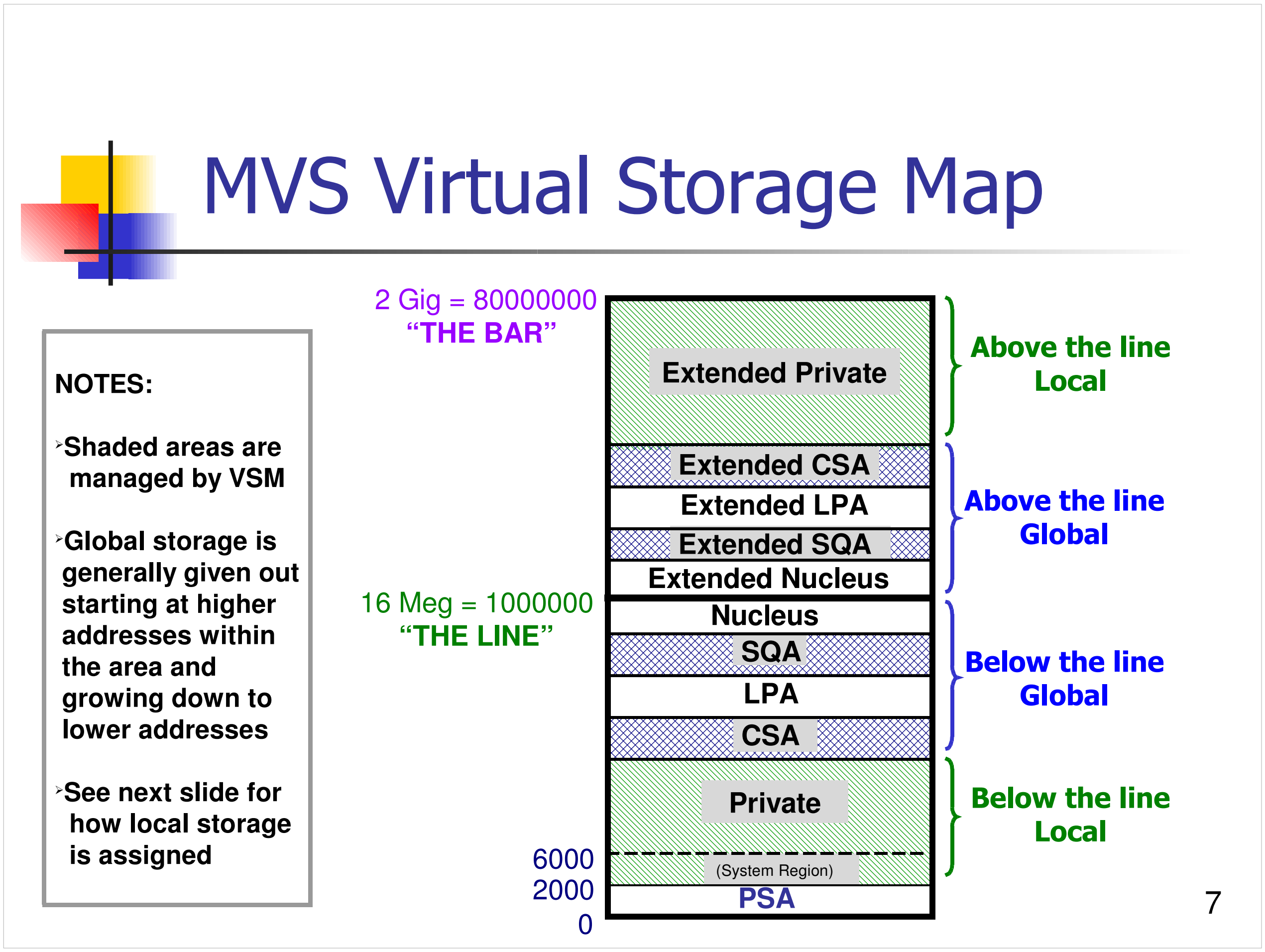
Areas of Virtual Storage

- Storage is subdivided into various areas
  - Global storage
    - Nucleus, Link Pack Area (LPA)
      - Not managed by VSM
    - Common Service Area (CSA), System Queue Area (SQA)
  - Local (or private) storage
    - User region
      - Available to unauthorized requesters
    - Local System Queue Area (LSQA)
    - High private

6

We have the concepts of global storage, which is common across all address spaces in the operating system, and local (or private) storage, which belongs to a particular address space and is readily accessible only by code running within that address space.

Global and local storage are further subdivided into storage areas of different characteristics and purposes.



BTL / **ATL** - Below the 16 Meg line / Above the 16 Meg line (but below 2Gig)

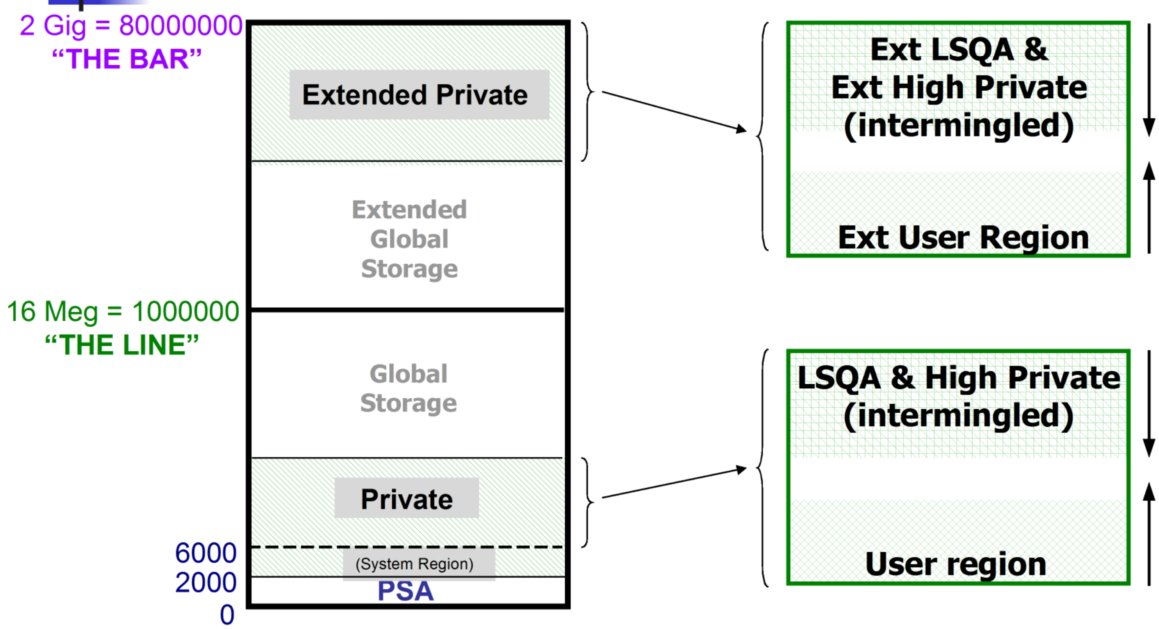
Extended storage = above the line storage

While there are small exceptions to this general rule, global storage is typically given out starting at the higher storage addresses and working its way down to lower addresses. This will become important later as we determine where to find the most recently allocated storage which is likely to have played a part in virtual storage exhaustion.

Note that the only global storage areas that VSM manages are SQA/ESQA and CSA/ECSA. The nucleus and LPA are also global storage, but modules permanently reside here. The storage is not available for any other use.

Through most of this presentation, we will not make a distinction between below the line storage and its extended above the line (extended) counterpart. For example, the term SQA will be used generically to refer to both SQA and ESQA.

Local Storage Management



8

Every address space has its own below the line and above the line private storage areas. The private storage area below the line and the private storage area above the line are each subdivided similarly into different subareas of storage. In either case, above or below, the private storage box has user region storage at the low end address range and LSQA/high private at the high end address range. As storage gets consumed, the two areas grow towards each other. If the two bump into each other, this is an out of local storage condition.

The separation between user region and LSQA/high private storage is intentional. User region storage is used by unauthorized programs. LSQA and high private storage are used by authorized programs. Therefore the two areas are kep segregated.

Note that <u>extended</u> private storage is **NOT** the same as <u>high</u> private storage.

Extended private storage is above the line private storage.

High private storage is another name for SP229, SP230, and SP249 storage; it gets allocated from the **HIGH** end of private or extended private storage

Local storage box = private storage area

User region = region = user private storage = low private storage

Global Storage Management

- CSA to SQA conversion
  - System will do all it can to honor a request for SQA
  - If no SQA is available, system will use free storage in the range of CSA to honor the request
  - CSA to SQA converted storage will be returned to CSA once the SQA occupying it is freed
  - Customers often rely on CSA-to-SQA conversion rather than waste storage through over-definition of SQA
    - This is okay in moderation
    - Risks include storage fragmentation, and inability to get SQA if CSA becomes exhausted

9

Global storage does not have the concept of two areas of storage growing towards each other as we saw is the case for local storage. The ranges of SQA and CSA storage are not contiguous with each other. Rather, SQA storage grows down within the defined range of SQA, and CSA storage grows down within the defined range of CSA.

However, global storage management does have a concept called CSA-to-SQA conversion. SQA storage is the virtual storage of choice for the most critical functions on the operating system. Therefore, VSM will do whatever it can to honor a request for virtual storage. If there is no more storage available in the range of SQA, VSM will convert a page or pages within the range of CSA to make it look like SQA storage, and will allocate this converted storage to the SQA requester. This storage will get converted back to CSA when it is freed.

The presence of CSA to SQA conversion on a system is not necessarily a problem. Some customers prefer to have a little conversion rather than having SQA storage sitting around unused.

Storage Management Implications

LOCAL

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

GLOBAL

- Excessive use of CSA can affect SQA on systems that rely on CSA to SQA conversion
- Excessive use of SQA can affect CSA

10

It's important to understand the give and take relationship between user region and LSQA/hi private, and between CSA and SQA. This means that while an ABEND878 or ABEND80A may have a reason code indicating that storage in a particular area could not be obtained, the fault could be tied to a misuse of storage in another area entirely. For example, if a program is in a loop obtaining storage in SQA, it will eventually exhaust SQA and the system will start converting CSA to SQA to honor the ongoing requests. Eventually storage in the range of CSA will become sufficiently depleted such that CSA requests cannot be honored either. This situation could result in abends indicating out of CSA and abends indicating out of SQA.

The only storage area whose growth can be limited is user region. This is because it is unauthorized. The system provides controls so that customers can specify limits on the growth of user region on a job by job basis. Such a limit would provide protection for LSQA and high private storage users in the address space. Note that if an address space begins suffering errors when trying to obtain LSQA storage, the address space is likely to memterm, whereas errors due to being out of user region tend to just lead to task termination. Abnormal memterms are considered highly undesirable in most customer environments. Therefore, limiting user region growth in high profile address spaces is a good thing.

Management of unassigned storage

- All pages of virtual storage are originally mapped to free pools
  - SQA pool, ESQA pool
  - CSA pool, ECSA pool
  - For each address space: PVT pool, EPVT pool
- Pages of storage are remapped from a free pool to a "subpool" as needed
  - Storage assigned to a subpool is said to be "allocated"
  - Storage assigned to a free pool is "unallocated"

11

Some people picture subpools as storage areas that have a pre-allotted amount of storage assigned to them, and once this storage is used, out of storage errors result. However, this perception is backwards. Subpools start out empty, and get storage assigned to them as storage requests request that storage for that subpool. Therefore, large subpools are indicative of a storage problem, not small subpools.

VSM holds unused pages of storage in free pools. There is a free pool for CSA below the line, CSA above the line, private storage below the line, and private storage above the line. These free pools are managed similarly. There is also a free pool of pages for SQA below the line and a free pool of pages for SQA above the line, but these pools are actually subpool-assigned to SQA subpool 245. These pages will get remapped to other SQA subpools when requests come in for those subpools. Once the storage is freed, these pages get reassigned to the free pool belonging to subpool 245.

Only full pages of storage reside in a CSA or private storage free pool. Only full pages of storage are remapped from a CSA or private storage free pool to a subpool. Only full pages of storage are remapped from the SP245 free pool to another SQA subpool.

What is a subpool?

- A subpool is a collection of pages of storage with related attributes
- Subpool numbers and attributes are defined by the operating system
  - Numbers range from 0 thru 255
  - Attributes include:
    - SQA, CSA, LSQA, high private, or user region
    - Pageable versus fixed
    - Fetch-protected or not fetch-protected
  - See MVS Diagnosis: Reference Chapter 8 for subpool numbers and attributes

12

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; SQA, CSA, LSQA, hi private, or user region. (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.

Storage Keys

- Every page of allocated storage has an associated key
  - Storage keys range from 0-15
    - System keys: 0-7
    - User keys: 8-15
  - The storage key offers protection for the page
- Storage key is determined by the storage requester
  - Sometimes implicitly defined by the subpool requested
    - SQA and LSQA storage are always KEY0
  - Sometimes specified by the requester of storage
    - CSA and high private
  - Sometimes determined based on the key of the job/program
    - User region

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.

Various Key Settings

- Key0 thru key7 are system keys (used by authorized code)
  - **Key0 – Used by operating system, has super authority**
  - Key1 – JES, APPC, TSO
  - Key5 – Data Management (DFSMS)
  - Key6 – VTAM, TCPIP
  - Key7 – IMS, DB2
- Key8 thru key15 are non-system keys
  - Key8 – This is the key most commonly used by unauthorized applications
  - Key9 – CICS users

14

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.

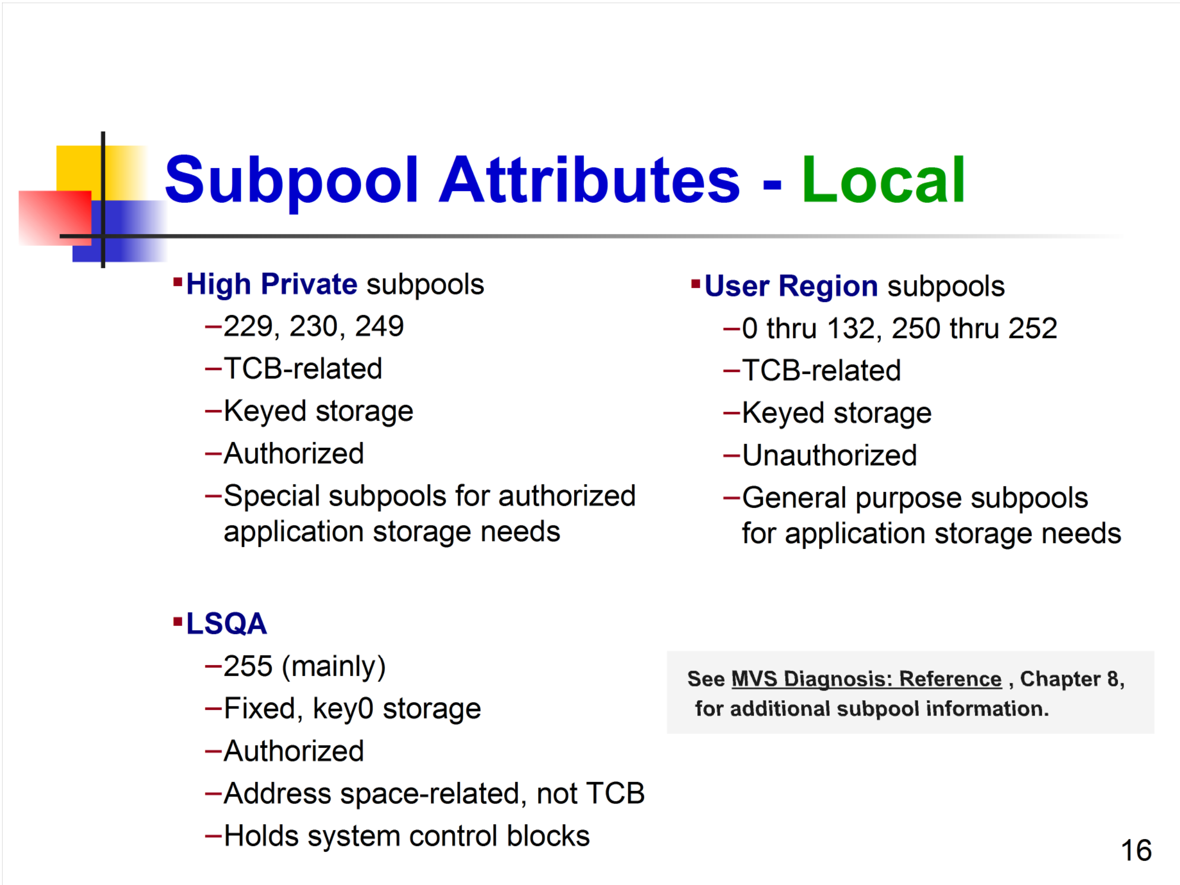
Subpool Attributes - Global

- **CSA** subpools
  - 227, 228, 231, 241
  - Keyed storage
  - Authorized
  - For authorized application and operating system needs e.g. JES, VTAM, DB2, OEM
- **SQA** subpools
  - 226, 239, 245, 247, 248
  - Fixed, key0 storage
  - Authorized
  - Used by BCP components of operating system

15

These subpool lists are not necessarily comprehensive, but they represent the most prevalently used subpools in the respective system storage areas. Note that use of both CSA and SQA requires authorization. Global storage is a limited resource, and exhaustion of it can affect everyone on the system. Therefore, VSM is not going to let programs with no authority (i.e. untrusted) request CSA or SQA.

SQA storage is fixed. That means it can be used by code that is running disabled. Disabled code cannot take interrupts, which includes I/O interrupts. If the program cannot taken an I/O interrupt, then it cannot use storage that could potentially get paged out.



On the left side of this slide are the authorized local storage subpools. There is some analogy here to the global storage subpools. LSQA is the local storage equivalent of SQA. It is always key0 and always fixed. It is used by critical operating system code. High private is somewhat analogous to CSA in that it 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 and for CSA.

VSM Storage Management Overview

- z/OS manages storage through the use of a variety of subpools designed to accommodate a variety of storage needs
  - Storage is allocated to a subpool in one page (4K) multiples
  - 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 (applicable to local storage only)
- When there is not enough storage above the line to fulfill an above the line storage request, VSM will attempt to honor the request from below the line
- LSQA / high private pages may not intermix with user region pages
- Generally VSM gives out storage at the high end of a page first

17

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.

An Example

- A program requests 2 pages of above the line SP230 Key1 storage
  - SP230 is private storage so 2 contiguous pages are taken out of the high end of the above the line private storage free pool in that program's address space
  - RSM sets up the 2 pages with the proper key
  - The 2 pages are assigned to subpool 230
  - The address of the beginning of the 2-page block is returned to the program

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

If the program had asked for one and a half pages of storage, this would still have resulted in 2 pages being reassigned from the free pool to the specified subpool. However, VSM would have then had to create an additional control block to show that half of one of the pages was free.

VSM Control Blocks

- **FBQE – free block queue element;**
  maps free pool pages, i.e. pages that are not subpool-assigned

For SQA/LSQA:

- **AQAT - address queue anchor table;**
  maps pages allocated to SQA/LSQA
- **DFE - double free element;**
  maps free storage within allocation represented by corresponding AQAT

For all other storage types:

- **DQE - descriptor queue element;**
  maps pages allocated to a specific subpool / key / TCB (as applicable) combination
- **FQE - free queue element;**
  maps free storage within allocation represented by corresponding DQE

Each of these 5 types of VSM control blocks have a similar construct, especially as VSM storage analysis is concerned. They have an ADDR field which denotes the address of storage that the control block represents. And they have a SIZE field which denotes the size of the storage this control block represents.

Viewing VSM Control Blocks

- 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
- To format VSM's global storage control blocks in a dump:
  - IPCS VERBX VSMDATA 'NOASID SUM'
    - NOASID can be abbreviated as NOA

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.” Similarly, instead of telling it to “show me global,” we tell it “don't show me ASID-specific storage.”

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)

```
   --FREE BLOCKS OF STORAGE

     FBQE: Addr 0003A000 Size      C000                                       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
```

- The **Addr** field defines the **beginning of a free block of storage**
  - Addr will always be **page-bounded**
- The **Size** field defines the **length of the free block of storage**
  - Size will always be 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 increments.

Example of AQATs/DFEs

```
AQAT: Addr 00ACA000 Size      1000                                       TCB:     n/a    SP/K: 255/ 0
                             DFE: Addr 00ACA000 Size       718           TCB:     n/a    SP/K: 255/ 0
AQAT: Addr 00AFC000 Size      2000                                       TCB:     n/a    SP/K: 255/ 0
                             DFE: Addr 00AFC000 Size        90           TCB:     n/a    SP/K: 255/ 0
                             DFE: Addr 00AFC240 Size        10           TCB:     n/a    SP/K: 255/ 0
                             DFE: Addr 00AFD5B8 Size        20           TCB:     n/a    SP/K: 255/ 0
```

- The page of storage at **ACA000** is assigned to **SP255**.
- Storage from **ACA000 thru ACA717** is free.
- **Therefore, storage from ACA718 thru ACAFFF is GETMAINed.**

- The **Addr** field of an **AQAT** defines the **beginning of an allocated block of storage**
  - Addr field of an **AQAT** will always be **page-bounded**

- The **Size** field of an **AQAT** defines the **length of the allocated block of storage**
  - Size field of an **AQAT** will always be a **multiple of a page** (X'1000')

- The **Addr** field of a **DFE** will fall within the **range of the associated AQAT**
  - The **Size** field of a **DFE** indicates the **length of the free area** of storage

An AQAT represents pages of allocated SQA or LSQA. Their corresponding DFEs describing free storage within the allocation are formatted underneath and to the right.

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

Example of DQEs/FQEs

```
Data for subpool 230, key  0 follows:

 -- DQE Listing (Virtual 31, Real 31)

  DQE:  Addr 7F606000 Size      1000                                        TCB: 00AFDD40 SP/K: 230/ 0
  DQE:  Addr 7F694000 Size      3000                                        TCB: 00AFDD40 SP/K: 230/ 0
                                    FQE: Addr 7F694000 Size       E8        TCB: 00AFDD40 SP/K: 230/ 0
                                    FQE: Addr 7F694D38 Size      768        TCB: 00AFDD40 SP/K: 230/ 0
```

- The **page** of storage at **7F606000** is assigned to **SP230 Key0**
  - All storage on the page is GETMAINed (no associated FQEs)
- The **3 pages** of storage beginning at **7F694000** are assigned to **SP230 Key0**
  - **Free storage: 7F694000 thru 7F6940E7; 7F694D38 thru 7F69549F**
  - **Getmained storage: 7F6940E8 thru 7F694D37; 7F6954A0 thru 7F696FFF**

- The **Addr** field of a **DQE** defines the **beginning of an allocated block** of storage
  - Addr field of a DQE will always be **page-bounded**

- The **Size** field of a **DQE** defines the **length of the allocated block** of storage
  - Size field of a DQE will always be a **multiple of a page** (X'1000')

- The **Addr** field of an **FQE** will fall within the **range of the associated DQE**
  - The **Size** field of an **FQE** indicates the **length of the free area** of storage

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.

VSMDATA 'NOGLOBAL SUMM' layout

- For each LSQA subpool
  - Oddly, the control block detail is broken into sections based on whether the virtual storage is backed by real storage below the line, above the line, or above the bar
    - AQATs/DFEs in ADDR order
    - Total allocated storage for section, broken into BTL and ATL
    - DFEs in SIZE order
  - Annoyingly, there is no display of total allocated storage per LSQA subpool in this section of the report
- FBQEs in ADDR order


```
Sections that subpool storage is broken down into are as follows:

Virtual 24, Real 24  (below the line virtual, backed by below the line real)
Virtual 24, Real 31  (below the line virtual, backed by above the line real)
Virtual 24, Real 64  (below the line virtual, backed by above the bar real)
Virtual 31, Real 31  (Above the line virtual, backed by above the line real)
Virtual 31, Real 64  (Above the line virtual, backed by above the bar real)
```

VSMDATA 'NOG SUMM' layout (cont)

- For each TCB
  - Subpool/key combinations used by TCB
    (subpool order: SWA, High Private, User Region)
    - As with LSQA, the control block detail is broken into sections based on whether the virtual storage is backed by real storage below the line, above the line, or above the bar
    - DQEs/FQEs in ADDR order
    - Total allocated storage for that subpool/key, broken into BTL and ATL
- Local storage map
- Summary of key fields from VSM Local Data Area (LDA)
- Summary of subpool totals, broken into BTL and ATL

When doing our local storage analysis, we will start with the items highlighted in red. These provide a high level view of storage usage within the address space. We will drill down from the map, to the subpool totals, to an individual subpool's control block detail.

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
```


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 5 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, that 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 multiple of the same subpool number but with different keys.

VSMDATA 'NOASID SUMM' layout

- For each SQA subpool
  - Again, the control block detail is broken into sections based on whether the virtual storage is backed by real storage below the line, above the line, or above the bar
    - AQATs/DFEs in ADDR order
    - Total allocated storage for section, broken into BTL and ATL
    - DFEs in SIZE order
  - Again, there is no display of total allocated storage per SQA subpool in this section of the report
- FBQEs in ADDR order

The format of the global storage report is highly analogous to that of the local storage report.

VSMDATA 'NOA SUMM' layout (cont)

- For each CSA subpool
  - Subpool/key combinations
    - As with SQA, the control block detail is broken into sections based on whether the virtual storage is backed by real storage below the line, above the line, or above the bar
    - DQEs/FQEs in ADDR order
    - Total allocated storage for that subpool/key, broken into BTL and ATL
- Global storage map
- Summary of key fields from VSM Global Data Area (GDA)
- Summary of subpool totals, broken into BTL and ATL

Again when we perform analysis we will start with the map and subpool summary, but these will be much less enlightening for global storage analysis than for local storage analysis.

Steps in diagnosing ABEND878 or ABEND80A

1. * Determine failing GETMAIN size/subpool
2. Review 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 no subpool/pattern found, check recently obtained storage and, for global storage, consider CSA tracker
7. If pattern cannot be ID'd, G/F trace
8. Absence of a pattern suggests tuning

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

These steps are similar for local and global storage analysis, but we will find that debugging local storage issues is a science while debugging global storage issues is an art.

Abend Information - ABEND878

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
     2-3   FFFFFFFF_00000602  FFFFFFFF_00000272
     4-5   FFFFFFFF_008E6968  FFFFFFFF_008E68E0
     6-7   FFFFFFFF_815AEC68  FFFFFFFF_00F57000
     8-9   FFFFFFFF_00000000  FFFFFFFF_000EF888
    10-11  FFFFFFFF_00000000  FFFFFFFF_7FFFDDC0
    12-13  FFFFFFFF_00000081  00000000_00006F60
    14-15  00000000_00007F3E  00000000_00000010
```

Abend Code

Third byte is Subpool number**

Length**

Return Code

**** Except if the fullword in Reg3 points to an ASCB control block!**
In this case, get subpool and length from linkage stack entry instead.

Except for the case of STORAGE OBTAIN, relevant error information can be obtained from the registers at time of error.

If Reg 3 is an ASCB address, then the abend was issued for a STORAGE OBTAIN request, and you will need to go to the linkage stack entry to gather details about the storage request that failed. There are several things that can be done to identify an ASCB address. First, an ASCB address always ends in X'00' or X'80'. Second, it is always below the line so will start with X'00'. ASCB's live in the range of SQA storage so the first signficant digit of the address will appear in the 3rd nibble and will typically be D, E or F. If you browse the storage, you will see an ASCB eyecatcher at +0.

The desired linkage stack entry should be the last (most recent) one under the TCB. In most cases, Reg 0 will contain the length of the request, and the third byte of Reg 15 will contain the subpool number.

You'll note that we put no emphasis on who did the GETMAIN or STORAGE OBTAIN that failed. This could be the bad guy, but much more often it is an innocent victim. It is good advice to do a thorough storage analysis rather than jump to a hasty conclusion based on the data at time of error.

Abend Information - ABEND80A

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 (80A, in register 1)

Subpool number in first byte; length in last 3 bytes (FA006000, in register 9)

Return Code (10, in register 15)

An ABEND80A is issued in response to an SVC A GETMAIN or its branch entry equivalent. This form of GETMAIN can only request storage below the line. Therefore, the length of storage requested will be X'FFFFFF' or less, i.e. you only need a max of 3 bytes to hold the length. Since a subpool number ranges from 0 thru 255, it can fit in one byte of storage. Therefore, for an SVC A GETMAIN, the subpool number and length fit into a single register.

The Reason Code

- Tells what area of storage is exhausted
  - 4 – SQA
    - Did SQA grow too large?
    - Or did CSA grow too large on a system that relied on available CSA to accommodate SQA overflow?
  - 8 – CSA
    - Did CSA grow too large?
    - Or did SQA overflow into CSA and squeeze it out?
  - C – LSQA or high private
    - Did LSQA/high private get too large?
    - Or did user region squeeze it out?
  - 10 – User region
    - Did user region get too large?
    - Or did LSQA/high private squeeze it out?

The return code identifies whether the out of storage abend is for local or global. As we can see here and will explore more later, the indication that a storage issue is with CSA versus SQA, or user region versus LSQA/high private can be ambiguous.

Let's go to the map!

- We'll start with a local storage example
  - 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 Rgn problem

```
                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
```


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!

- Another local storage example
- 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
  - LSQA size usually <1Meg
  - ELSQA size usually <X'40'Meg
- **Conclusion: User Rgn problem**

```
                    LOCAL STORAGE MAP
  ___________________________
 |                           |80000000  <- Top of Ext. Private
 | Extended                  |
 | LSQA/SWA/229/230          |80000000  <- Max Ext. User Region Address
 |___________________________|7F5C3000  <- ELSQA Bottom
 |                           |
 | (Free Extended Storage)   |
 |___________________________|7F580000  <- Ext. User Region Top
 |                           |
 | Extended User Region      |
 |___________________________|25D00000  <- Ext. User Region Start
 :                           :
 : Extended Global Storage   :
 =======================================<- 16M Line
 : Global Storage            :
 :___________________________:  900000  <- Top of Private
 |                           |
 | LSQA/SWA/229/230          |  900000  <- Max User Region Address
 |___________________________|  8C5000  <- LSQA Bottom
 |                           |
 | (Free Storage)            |
 |___________________________|  878000  <- User Region Top
 |                           |
 | User Region               |
 |___________________________|    6000  <- User Region Start
 : System Storage            :
 :___________________________:       0
```


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

```
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)
```

Total storage Allocated to BTL User Region (pointing to LOAL)

Total storage Allocated to BTL LSQA/high private (pointing to HIAL)

Note: ATL equivalent fields are also reported

- From the storage map, you can easily calculate the **difference** between the **bottom/top of user region** and the **bottom/top of LSQA/high private.**
- **LOAL** and **HIAL** report how much storage is allocated in each of those areas.
- A large disparity would suggest fragmentation. Check FBQEs for more info! (But 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

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

- User region subpools: 0-132, 250-252
- Is there a subpool that looks overgrown?
  - Locate subpool's control blocks in the upper section of the report
- If nothing jumps out, 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 the user region subpool's control blocks

```
-- DQE Listing (Virtual 31, Real 31)

 DQE:   Addr 25D00000 Size     F0000
                                     FQE: Addr 25D00000 Size        778
 DQE:   Addr 25DF0000 Size     F0000
                                     FQE: Addr 25DF0000 Size        778
-   - - - - - - - - -   3,044 LINE(S)  NOT DISPLAYED  - - - - - - - -
 DQE:   Addr 7F1C0000 Size     F0000
                                     FQE: Addr 7F1C0000 Size        778
 DQE:   Addr 7F2B0000 Size     F0000
                                     FQE: Addr 7F2B0000 Size        778
 DQE:   Addr 7F3A0000 Size     F0000
                                     FQE: Addr 7F3A0000 Size        778
 DQE:   Addr 7F490000 Size     F0000
                                     FQE: Addr 7F490000 Size        778

***** Subpool    2, key  8 Total alloc:  5A0F0000 ( 870000 Below, 59880000 Above )
```

- **F 5A0F0000 FIRST**
  - Brings you to subpool total line
- 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

1. * Determine failing GETMAIN size/subpool
2. Review 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 no subpool/pattern found, check recently obtained storage and, for global storage, consider CSA tracker
7. If pattern cannot be ID'd, G/F trace
8. Absence of a pattern suggests tuning

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

39

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?

- 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: **F 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

40

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 suspect subpool is found?

REPORT SORTING TECHNIQUE

- While viewing a VSMDATA SUMM output under IPCS:
  - **SORT 115 122**
  - **F 'SP/K' FIRST** to get to the relevant lines
  - If desired, a block delete can be done to remove all lines preceding the first relevant line

- **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 we will see that 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.

What if storage is creeping but has not abended yet?

- 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.

What if I can't figure out who is responsible for a pattern?

GETMAIN / FREEMAIN/STORAGE trace

- Used when pattern is found but cannot be identified
- Requires recreate or 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.

What about global storage?

- Global storage analysis follows the same steps as local
- Differences
  - Use: **VERBX VSMDATA 'NOA SUMM'**
  - Additional diagnostic tool: "CSA Tracker"
  - Typical doc is a console dump rather than an abend dump (could even be a SADump)
  - Trickier diagnosis
    - Global map not as useful; more reliance on FBQEs
    - Subpool patterns are more subtle
  - Greater reliance on comparison dumps
  - Often related to system tuning

While the party line is that you can follow the same steps to debug global storage problems as you use to debug local storage problems, the fact is that identifying which subpool is overgrown can be a lot more difficult for global storage. A comparison dump that shows the system when its storage usage is healthy can make a big difference. Another help is when two dumps can be compared, the first showing creep and the second showing more creep.

Whereas in local storage analysis we didn't need to pay a lot of attention to FBQEs, with global storage analysis, FBQEs help us understand whether the problem is above or below the line.

We also will need to consider whether CSA to SQA conversion is taking place and how that fits into our picture.

Global storage map

- Now we'll do a global storage example
  - IP VERBX VSMDATA 'NOA SUM'
- Report only shows global storage definitions
  - Does not show growth the way the local report did
- Instead refer to:
  - CSA to SQA conversion info
  - FBQEs

```
                  GLOBAL STORAGE MAP
 ___________________________________ 80000000  <- Top Of Storage
:                                   :
:  Extended Private Storage         :
:___________________________________:25D00000  <- Ext. CSA Upper Bound
|                                   |
|  Extended CSA                     |
|___________________________________| 68E1000  <- Ext. CSA Lower Bound
|                                   |
|  Extended PLPA/FLPA/MLPA          |
|___________________________________| 2711000  <- Ext. SQA Upper Bound
|                                   |
|  Extended SQA                     |
|___________________________________| 1A31000  <- Ext. SQA Lower Bound
|                                   |
|  Extended Nucleus                 |
|                                   |
=======================================<- 16M Line
|  Nucleus                          |
|___________________________________|  FD7000  <- SQA Upper Bound
|                                   |
|  SQA                              |
|___________________________________|  D70000  <- SQA Lower Bound
|                                   |
|  PLPA/FLPA/MLPA                   |
|___________________________________|  BAB000  <- CSA Upper Bound
|                                   |
|  CSA                              |
|___________________________________|  900000  <- CSA Lower Bound
:  Private Storage                  :
```


We show the map mostly as demonstration. This map has little to offer when doing global storage analysis, although it is quite useful when trying to understand whether an address that you have encountered in general debugging is a private, CSA, LPA, SQA, or nucleus address.

CSA to SQA conversion info

- Located just below global storage map

```
Summary of Key Information from GDA (Global Data Area) :

   SQA:  Address Range =>     E6A000    -    FD5000   Size =>  16B000
   CSA:  Address Range =>     800000    -    B77000   Size =>  377000

   ESQA: Address Range =>    1BB7000    -   6CB0000   Size => 50F9000
   ECSA: Address Range =>    9F7F000    - 28800000   Size =>1E881000

   CSA   Converted to SQA: Below 16M =>         0
                           Above 16M =>    DE7000
                           Total     =>    DE7000
```


When debugging a global storage issue, note whether CSA to SQA conversion is occurring, both below and above the line. A RC8 abend (CSA) with no conversion would imply the storage growth is solely a CSA problem.

In the example on the slide, there is no CSA to SQA conversion occurring below the line. Above the line, nearly 14Meg has been converted. Given how big ECSA is, this amount of conversion probably isn't enough to cause a system to start suffering global storage abends. However, we would want to look at FBQEs before drawing conclusions.

What's free in CSA?

```
CSA (Below) Region Descriptor data follows

   --Start Addr 00800000 Size   377000

   --FREE BLOCKS OF STORAGE

     FBQE: Addr 00800000 Size   2A4000
     FBQE: Addr 00ABD000 Size     7000

CSA (Extended) Region Descriptor data follows

   --Start Addr 09F7F000 Size 1E881000

   --FREE BLOCKS OF STORAGE

     FBQE: Addr 09F7F000 Size     2000
     FBQE: Addr 09F86000 Size     1000
```

- FIND FBQE from top of report
- How much CSA is free BTL?
  - Quite a bit
- How much CSA is free ATL?
  - Not much! (Since global storage grows down, biggest free blocks will be at lowest addresses)

This is from the same dump as the previous slide. In this example, we have an FBQE showing X'2A4000' bytes of free storage below the line at the bottom of CSA, but when we look at the above the line picture, we see very little free storage at the bottom of ECSA. ECSA is effectively exhausted.

We could have an overgrown ESQA (above the line only) subpool.

Alternatively, we could have a case where the problem is with a CSA subpool, and its growth has exhausted ECSA and is now chipping away at CSA.

We saw on the previous slide that there was CSA to SQA conversion above the line, so we know ESQA requests are depending on available ECSA to be successfully honored. It is also true that there are a couple of SQA subpools that are above the line only pools. A request for such a subpool under these conditions would result in an out of SQA abend – RC4.

ST REGS against this dump shows R1=84878000, R15=00000004, and R3 having a X'F8' = 248 in its 3rd byte. SP248 is an above the line only ESQA subpool.

We'll come back to this.

What's free in SQA?

- Free BTL SQA is mapped to an BTL DFE in SP245
- Free ATL SQA is mapped to an ATL DFE in SP245
- If you really need to find these DFEs, check out the end of the SP245 DFE chain that is formatted out in SIZE order
  - This step can usually be avoided!

This is not a step you want to take unless you really have to. Not only can it be challenging to navigate the AQATs and DFEs to the appropriate size-order section, but sometimes the DFE chain changes while the dump is being taken, making it appear “broken” so you can't get to the end of it.

Thinking about what we know

- RC4 dump – out of SQA (so out of CSA as well)
  - FBQEs will tell us if:
    - We are out of BTL global only
    - We are out of ATL global only
      (some SQA subpools are ATL only!)
    - We are out of both BTL and ATL global
- RC8 dump – out of CSA
    - FBQEs will tell us if we are out of BTL CSA or all CSA
    - If CSA to SQA conversion, then SQA users also impacted
- In either case, use the report sorting technique to sort the global storage control blocks into ascending ADDR order
  - Look for patterns at the bottom (the most recently allocated storage) of the range of CSA that is exhausted

The example that we have been looking at is from an ABEND878 RC04 dump. Failing subpool is SP248 which is an above the line only SQA subpool.

Conversion shows us that we are relying on CSA to SQA conversion for our above the line SQA needs. If problem were truly with an SQA subpool, we would expect to see a lot more conversion.

FBQEs show we are out of ECSA above but still have plenty of CSA below.

We could have a case where the problem is with a CSA subpool, and its growth has exhausted ECSA and is now chipping away at BTL CSA. We know ESQA requests are depending on available ECSA to be successfully honored, but now ECSA is full.

Continuing with our example

- Use the report sorting technique

```
FBQE: Addr 09F7F000 Size      2000                                     TCB:     n/a    SP/K: n/a
DQE:  Addr 09F81000 Size      5000                                     TCB:     n/a    SP/K: 231/ 7
                                   FQE: Addr 09F81000 Size      F00    TCB:     n/a    SP/K: 231/ 7
FBQE: Addr 09F86000 Size      1000                                     TCB:     n/a    SP/K: n/a
DQE:  Addr 09F87000 Size      5000                                     TCB:     n/a    SP/K: 231/ 7
                                   FQE: Addr 09F87000 Size      F00    TCB:     n/a    SP/K: 231/ 7
DQE:  Addr 09F8C000 Size      5000                                     TCB:     n/a    SP/K: 231/ 7
                                   FQE: Addr 09F8C000 Size      F00    TCB:     n/a    SP/K: 231/ 7
DQE:  Addr 09F91000 Size      9000                                     TCB:     n/a    SP/K: 241/ 6
DQE:  Addr 09F9A000 Size      5000                                     TCB:     n/a    SP/K: 231/ 7
                                   FQE: Addr 09F9A000 Size      F00    TCB:     n/a    SP/K: 231/ 7
DQE:  Addr 09F9F000 Size      5000                                     TCB:     n/a    SP/K: 231/ 7
                                   FQE: Addr 09F9F000 Size      F00    TCB:     n/a    SP/K: 231/ 7
```

Note the pattern of 5000/F00 for the SP231 KEY7 DQE/FQE blocks.

Here we see a pattern using the report sorting technique. SP231 is a CSA subpool. If filled ECSA and now presumably will continue to fill BTL CSA. But before it got too far, the system got burned with a request for SP248 SQA (ATL only subpool) that could not be honored because the system was relying on CSA to SQA conversion above the line, and ECSA was effectively exhausted.

Storage creep rather than abend?

- **Console dump – storage still available but creep detected**
  - **Did monitor show a CSA creep? BTL or ATL?**
    - Check subpool summary for overgrown CSA subpools
  - **Did monitor show an SQA creep? BTL or ATL?**
    - Check subpool summary for overgrown SQA subpools
    - Problem growth in SQA SP245 may not be visible in the subpool summary due to how free SQA is managed,
      i.e. SP245 will always look large
  - **Did monitor show a specific subpool growing?**
    - Go straight to the control blocks for that subpool
  - **Note:** Normal global subpool sizes can be highly variable
    - May be difficult to identify which subpool is overgrown
      - Find comparison dump from same system when healthy
      - Or take a console dump after additional growth

In the case of a global storage creep, it is possible to debug the problem with just one dump but having a comparison dump would make things much easier.

CSA Tracker

- Reports global storage (both CSA and SQA) ownership by job or by individual storage requests
  - **IP VERBX VSMDATA 'OWNCOMM'**
    - Summary report
    - For each job/ASID that owns global storage on the system, reports amount of CSA, ECSA, SQA, and ESQA owned as well as total
  - **IP VERBX VSMDATA 'OWNCOMM DETAIL SORTBY(xxxxxxxx) CONTENT(NO)**
    - For each "getmained" piece of global storage, reports start address, size, who obtained it, and time it was obtained
    - Sortable by time, address, length, etc
    - Removing 'CONTENT(NO)' will provide snippet of first 4 words of data in each storage area

CSA tracker was designed to help the debugger identify who owned a piece of SQA or CSA storage. However, it can also be a useful tool in tackling global storage growth issues. Here we will provide an overview of the report's most useful features, and leave it as an exercise for you to imagine how you might apply to global storage analysis.

Excerpt from CSA Tracker Detail Report

IP VERBX VSMDATA 'OWNCOMM DETAIL SORTBY(ADDR) CONTENT(NO)'

```
ASID Job Name Id        St T Address  Length   Ret Addr MM/DD/YYYY HH:MM:SS
---- -------- -------- -- - -------- -------- -------- ---------- --------
0380 IMSL     S0142756 Ac C 00AB79D8 000002C0 25710C88 05/22/2014 10:24:11
0265 DBWEDBM1 S0061820 Ac C 00AB8000 00001000 2880C6B8 05/21/2014 12:05:37
0000 *SYSTEM* ........ Ac C 00AB9000 00000FF0 0000C664 05/21/2014 12:05:28
0380 IMSL     S0142756 Ac C 00ABA000 00001000 25710C88 05/22/2014 10:24:09
0000 *SYSTEM* ........ Ac C 00ABB000 00000FF0 0000C664 05/22/2014 09:54:29
0224 IMSL     S0061576 OG C 00ABCFE0 00000020 0028DBC2 05/21/2014 12:05:17
```

Owning addr space Gone or Active

CSA or SQA

Address of requester

Time requested

Note that individually getmained areas of storage are detailed here. If the requesting code for a particular control block lives in global storage, you can likely use this same dump to identify specifically who the requester was. However, if the requester of a piece of global storage did so from code resident in private storage, you likely won't have that address space in the present dump and so would not be able to use this dump to determine who made the request. While it's not guaranteed that the 'Ret Addr' is in the address space indicated under the ASID column, odds are good that this is the case. Sometimes taking a console dump of the indicated job/ASID will still allow successful mapping of the private storage Ret Addr. If the address points immediately after an SVC 78, BALR, or PC instruction, then you're probably in luck.

'Owner Gone' means the address space has gone away and left this piece of CSA or SQA storage behind. Sometimes this is intentional, such as when an address space comes up at IPL time, puts hooks and programs into place in common interfaces, then goes away. Other times the storage has been left around by mistake.

Excerpt from CSA Tracker Summary Report

IP VERBX VSMDATA 'OWNCOMM'

```
******************************** GRAND TOTALS ********************

                          Total
Description               Length   SQA    CSA    ESQA     ECSA
------------------------  -------- ------ ------ -------- --------
Total SYSTEM-owned        072A1C18 11A918 014590 048044C8 0296E8A8
Total for active ASIDS    18FFDF20 01B4F8 09FDC0 01164880 17DDE3E8
Total for "Owner Gone"    00211688 0029F8 000158 00006628 00208510
Total for "No Detail"     FFFFDDC0 000000 000000 FFFFDDC0 00000000
                          -------- ------ ------ -------- --------
Grand Total               204AEF80 138808 0B44A8 0596D130 1A9551A0

*******************************************************************

                            Total
 ASID Job Name Id        St Length   SQA    CSA    ESQA     ECSA
 ---- -------- -------- -- -------- ------ ------ -------- --------
 0000 *SYSTEM* ........ Ac 072A1C18 11A918 014590 048044C8 0296E8A8
 0001 *MASTER* ........ Ac 01DA2098 00D058 030D90 00496878 018CDA38
 0002 PCAUTH   ........ Ac 000007E8 000040 000000 000007A8 00000000
 0003 RASP     ........ Ac 000005C0 000000 000000 000005C0 00000000
```


The VSMDATA OWNCOMM Summary report is convenient for getting an overview of storage usage by specific address spaces on the system. IPCS sorting techniques can be used to sort the various address space entries into an order other than by ASID number.

Common VSM Problems

- ABEND40D RC10 memterm dump has no useful storage
  - Sometimes address space terminates before dump completes
  - Most ABEND40D's are preceded by ABEND80A's and/or ABEND878's
    - **Set SLIPs for COMP=878 and COMP=80A in problem job**
    - **If no limit on user region, consider setting one**
- After IPL, many jobs suffering ABEND878's/ABEND80A's
  - Probably lost a meg of private storage due to global storage definitions increasing
    - **Health Check CHECK(IBMVSM,VSM_CSA_CHANGE)**
    - **Health Check CHECK(IBMVSM,VSM_PVT_LIMIT)**

If setting SLIPs and limiting user region growth still does not allow a complete dump to be taken, contact VSM L2 for assistance. They likely will want to view the IEF374I message in the job's joblog, then use this data to set a SLIP which will trigger a dump when LDALOAL, LDAHIAL, LDAELOAL, or LDAEHIAL exceeds a particular threshold.

VSM_CSA_CHANGE checks at IPL time for changes in the size of CSA or private (including the extended areas) since last IPL.

VSM_PVT_LIMIT checks at IPL time for the size of private storage or extended private storage falling below a specified size.

Common VSM Problems

- ABEND822 abending jobs on initiator
  - **Problem:** Job leaves LSQA storage behind, preventing the next job from getting the desired user region size
  - **Solution:** Option **CHECKREGIONLOSS in DIAGxx** parmlib member
    - Indicates the amount of region size loss that can be tolerated in an initiator address space
    - At termination of each job run in the initiator, if the maximum available region size has decreased from the initial value by more than the CHECKREGIONLOSS specification, the initiator terminates with message IEF093I
    - JES2, JES3, WLM, and APPC all automatically restart initiators
    - See MVS: Initialization and Tuning Reference for more info

Common VSM Problems

- "Natural" CSA storage fragmentation, especially in IMS environments
  - VSM APAR OA31115 introduced a new **DIAGxx parmlib option: BESTFITCSA(YES|NO)**
    - VSM implements a "best fit" algorithm for CSA requests rather than "first fit"
    - Additional VSM cycles required to locate best fit
    - See MVS: Initialization and Tuning Reference for more info

IBMVSM Health Checks

- VSM_CSA_LIMIT
- VSM_SQA_LIMIT
- VSM_PVT_LIMIT

One time check at IPL to ensure that the size of the indicated storage area is above a specified minimum

- VSM_CSA_THRESHOLD
- VSM_SQA_THRESHOLD

Check made on an interval to ensure that the amount of available CSA/SQA has not fallen below a specified threshold

- VSM_CSA_CHANGE – check for changes in the size of CSA or private (including extended areas) since the last IPL
- VSM_CSA_LARGEST_FREE – check to ensure largest CSA/ECSA free area is above a specified size

VSM_CSA_THRESHOLD, VSM_SQA_THRESHOLD, and VSM_LARGEST_FREE all support dynamic severity checking.

There is one other VSM healthcheck: VSM_ALLOWUSERKEYCSA. This check issues a warning if the setting of DIAGxx parameter ALLOWUSERKEYCSA is YES.

See the IBM Health Checker User's Guide for more information.