INTERCOMM

STORAGE MANAGEMENT ENHANCEMENTS

Introduction

Two major enhancements have been made to storage management processing. They are support for 31-Amode pools, and pool header integrity checking and overlay recovery. To provide this new support, MANAGER and associated storage management modules have been substantially modified and processing overhead has been minimized.

31-Amode pools can be used with the existing 24-Amode pools, because the 31-Amode pools module is loaded at startup (if defined), and it's pointers are then resolved into previously reserved fields in the SPAEXT. Statistics for tuning of the 31-Amode pools are similar to those for 24-Amode pools.

Pool header integrity checking and recovery is designed to prevent unexpected program checks in MANAGER, and to provide easier debugging of a core (pool header) overlay problem. In addition, RMINTEG processing has been redesigned (will not abend the system).

Support for these Intercomm enhancements are incorporated in Release 11 and are provided for Release 10 at SM level 2300 when Experimental SMs 871-874, 879, 896, 902, and 914 have been applied. For Release 10, all of the EX's must be applied, even if 31-Amode pools are not needed.

The following sections describe the new storage management support and include:

- Overview
- Installation
- Macro Changes
- New/changed Messages
- New/changed MANAGER ISK Codes
- New Snaps.
- Internals Changes

Documentation changes will be incorporated in future updates to the following manuals:

Basic System Macros
Assembler Language Programmers Guide
Messages & Codes
Operating Reference Manual
Table Facility.

OVERVIEW

1) 31-Amode Pools Support.

31-Amode pools can be used to satisfy Intercomm STORAGE macro requests with LOC=ANY (described in Basic System Macros) which do not force an IBM GETMAIN (via BOUND=PAGE, the SP (non-zero), or POOLS=NO parameters).

In the same manner as for 24-Amode pools, a 31-Amode pools module is coded using the COREACCT macro for tuning statistics, and ICOMPOOL macros to define the number and size of the desired pool blocks. The load module name must be IC31PLnn, which is loaded at startup if POOLSTRT is included in the Intercomm linkedit. IC31PLnn must be prelinked with the RMODE=ANY and AMODE=31 parameters to force loading above the 16M line. To define the load module suffix (nn), a new parameter on the SPALIST macro (IC31PL=nn/NO) must be coded. If the IC31PL parameter is not NO, POOLSTRT attempts to load the IC31PL module with the coded 2-byte alphanumeric suffix and issues a message (MI071R) if unsuccessful/load module not 31-Amode, or resolves entry-points into the SPAEXT if successful.

TDUMP (Thread Resource Dump report processing) and SYSCNTL (SCTL$TDUMP command) have been modified to display 31-bit pool addresses, where appropriate.

INTTABLE has been modified to use the 31-Amode pools, if present and desired, for Table (and Page) Facility processing: ensure one 96-byte pool is defined for table access snap processing (if activated). If the 31-Amode pools are only used for Table/Page Facility processing, then define the other pools in increments of 1 or more K (depending on initial and increment values defined on the SPALIST macro for the TFCB and table areas). See Table Facility for further details: note that a GETMAIN is forced for these areas if the TABSP parameter is coded with a non-zero Subpool number.

2) Storage Management Processing Changes.

MANAGER has been completely rewritten to also process the 31-Amode pools (if loaded successfully). Because 31-Amode storage request processing requires RCB's, the RM global has been removed from INTGLOBE and SETGLOBE and the RCB Table area is always acquired. In addition, MANAGER now tests for the presence of 24- or 31-Amode pools, as appropriate for the LOC parameter on the STORAGE macro, and acquires pool storage accordingly if available (and GETMAIN not forced). Therefore, the RMPOOLS global has also been removed.

Core Use Statistics (if RMSTATS global set to 1) have been expanded to report both 24- and 31-Amode storage/pool requests. If implemented (RMACCT global set to 1), Request Distribution statistics and Pool Detail statistics are provided for both 24- and 31-Amode pools, if present. RMTRACE has been rewritten to provide these tuning statistics (note that Distribution lines for zero requests are omitted to save print space). POOLDUMP has also been optimized to use less print lines/processing time, and modified to print 31-Amode pool headers as for 24-Amode pools. POOLDUMP will recognize an invalid pool header and prints it in Hex if no RCB is found for its address.

OVERVIEW, con't.

3) User Interface Changes.

User Assembler programs loaded above the 16M line (executing in 31-Amode) may use the SPA parameter on the STORAGE and STORFREE macros, and the SPAEXT parameter on the PASS and CATCH macros. That is, they may now enter MANAGER in 31-Amode. MANAGER (RSMGMNT Csect) will then execute in 24-Amode (except when processing 31-Amode pools/storage), but will return in 31-Amode to a caller executing in 31-Amode (above or below the 16M line) when one of the above macros was issued.

If the SPA parameter is used on a STORAGE macro by a 31-Amode program, the LIST parameter may be omitted (and RENT=NO is forced). If LIST is used, then it must point to 24-Amode storage (high-order byte of LIST address must be zero: MANAGER will force a program check if it is not).

If LOC=ANY is coded on the STORAGE macro, all programs may also use the ADDR parameter to point to a 31-Amode address. Otherwise, omit the ADDR parameter or ensure it is a 24-Amode address (high-order byte is zero), especially if requesting storage with register notation for the ADDR parameter. If LOC=ANY was not coded for the STORAGE macro, then MANAGER will issue new message RM029A and force a program check if the ADDR parameter is coded for a 31-bit address (high-order byte not zero).

On a STORFREE, the user must ensure the high-order byte of the address passed for the ADDR parameter is zero if trying to free 24-Amode storage. Note that a 24-Amode storage address returned by a STORAGE macro will always have a high-order zero byte.

BATCHPAK has also been modified to allow STORAGE and STORFREE request entry and exit in 31-Amode, and to allow the revised INTTABLE (for Table Facility) to execute in batch mode. The LOC=ANY parameter on a STORAGE causes 31-Amode core to be acquired, while a non-zero hi-order byte on the address of storage to be freed indicates 31-Amode core to STORFREE.

4) Pool Header Integrity Checking and Recovery.

To prevent problems due to overlaid pool headers (in the appropriate 24- or 31-Amode pool size area), MANAGER validates the 8-byte pool header for STORAGE (first free block header) and STORFREE requests. If an invalid header is found, a message (revised RM012A) is issued giving the header's address, and a snap (ID=92 if STORAGE, ID=93 if STORFREE) of the ICOMCHN Csect (pool area pointers) and of the invalid header's pool area is provided. On a STORAGE, if the header of the first free block (if any) is invalid, the free chain pointer (in ICOMCHN) is then zeroed, and a GETMAIN for the requested storage is done. On a STORFREE, if the low-order 4 bytes (2^{nd} word) of the header of a block being freed is invalid, the block is not returned to the pool's free chain (only the RCB is freed). If only the first 2 bytes (count of bytes in use) in the header is invalid, the count is then reset to the maximum block size for that pool area, and STORFREE processing continues. Note that the second halfword of the header of a block in use contains the associated RCB offset, and if invalid, MANAGER searches the RCB Table for the corresponding RCB, therefore it does not need validating.

Before return to the caller, a Thread Resource Dump is produced for user debugging (current owner of previous block may have done the overlay). Thus, program checks (S0C4) in MANAGER that eventually bring down the system are avoided.

OVERVIEW, con't.

Although some (or eventually all) of the blocks in a pool area with invalid headers may be lost for the duration of Intercomm execution (and more GETMAIN and FREEMAIN processing than normal is executed), at least a header overlay will not eventually cripple the system/cause an abend. Should the pool block following the invalid header also have been overlaid, and it is currently in use (message RM012A indicates STORFREE found the invalid header), it is probable that processing integrity for the owner issuing the STORFREE was compromised, and it may be necessary to bring down the system anyway. Immediate examination of the snap can determine the extent of the overlay and the appropriate action.

In addition, another source of program checks (in thread 0) in MANAGER is avoided in the 'averaging' routine (Entry Point AVRAGING) for pool use detail statistics calculation of the average free blocks per pool size. This routine chains down the free pool block chain for each size in both 24- and 31-Amode pools (if present) and now checks the validity of the forward pointer before proceeding to the header of the next free block. If the pointer has been overlaid, message RM012A is issued and a snap (ID=91) of the ICOMCHN and ICOMPOOL Csects, and a Thread Resource Dump, are provided for debugging. Note that the frequency of executing the averaging routine is controlled (as before) via the RMSTIM parameter on the SPALIST macro and therefore, this verification routine may find a bad header before a STORAGE request for the associated pool size, increasing the possibility of finding the offender.

Revisions to MANAGER have also changed the RMINTEG option processing to just validate the header of the next higher pool block on a STORFREE from the 24-Amode pools. RMINTEG is only called if the header of the block being freed is valid. Thus, if the header of its neighbor pool block is invalid, it is highly likely that the STORFREE issuer caused the overlay. When an invalid neighbor header is found, revised message RM022A is issued. Then, RMINTEG processing is temporarily disabled while a snap (ID=94) of the message, RMSAVE area (STORFREE caller's registers), ICOMCHN and ICOMPOOL Csects and a Thread Resource Dump are provided for debugging (during this process, both MANAGER's and the RMSAVE registers are temporarily saved in a GETMAIN'ed area used as a save area by RMINTEG). To preclude the invalid neighbor pool header from causing multiple snaps, RMINTEG attempts to correct it by making the header the end of the free chain (if it was found on the area's free chain), or by fixing it if an RCB could be found for it (pool block is in use). If the last block in the ICOMPOOL Csect is being freed, then RMINTEG validates the POOLEND Csect, and corrects it if overlaid. If multiple snaps occur for the same pool block header (see also message RM012A), then it will be necessary to bring down the system and debug the problem. This option should be activated (via STRT command) in the production system when the earlier described header checking and/or TRAP implementation (see below) does not locate a random core clobber in the pools. It should always be active in test environments to prevent future production problems. Also, in test, the 24-Amode pools module should mirror the production system, but with adjustments for new message and save area (or DWS, or ISA) sizes, etc., as needed. For this option, set the RMINTEG global to 1 for MANAGER assembly.

TRAP has been optimized for header and chain checking and modified to permit the user to define a specific pool size for checking only the free chain and block headers of that pool's area. User code may be easily inserted to test for a known clobber (such as a X'40' (blank) in the first byte of a header) in all headers or blocks, or only in those of a specific pool size. See the comments in the revised TRAP on how to implement either, or both, of these overlay trapping options. Also, TRAP (after disabling itself) now goes through standard STAEEXIT processing when it issues its Abend 1369 (snap 122 and Thread Resource Dump issued, etc.). A TRAP31 version (of the revised TRAP) for the 31-Amode pools has also been created (both modules may not be used in the same link).

INSTALLATION

After installing Release 11 or applying the Experimental SMs to Release 10 for the storage management enhancements and assembling and linking the associated modules and the 24-Amode pools tables for all regions, do the following steps to implement the new support:

- If desired, code and assemble a 31-Amode pools table using the COREACCT and ICOMPOOL macros as for a 24-Amode pools table (use a Csect name of COREACCT, not ICOMINX), or modify the provided sample IC31PL00, and give it a load module name of IC31PLnn while linkediting it with the parameters RMODE=ANY and AMODE=31

- If an IC31PL load module is created, define its 2-byte suffix value for the SPALIST macro using the new IC31PL parameter and reassemble and link the SPA table

- If a 24-Amode pools module (ICPOOLnn) is to be loaded, ensure that a COREACCT CSECT statement is before the COREACCT macro, omit any other CSECT names.

- If 24- and/or 31-Amode pools tables are to be loaded, add an INCLUDE for the revised POOLSTRT module to the Intercomm linkedit control statements (with startup modules) - do not ORDER POOLSTRT and do not include IC31PLnn

- If the 24-Amode pools module is in the Intercomm linkedit, add ORDER statements for the COREACCT, POOLCONS, and POOLACCT Csects before those for ICOMINX/ICOMCHN/ICOMPOOL/POOLEND, if not already coded: it is required that the ICOMCHN Csect immediately precedes the ICOMPOOL Csect and that the POOLEND Csect immediately follows the ICOMPOOL Csect

- Linkedit the Intercomm load module (after defining the library containing the modules and tables revised by the EX's as first in the SYSLIB concatenation)

- Relink dynamically loadable Assembler programs linked with INTLOAD on the DYNLLIB library (ensure the revised (by EX 879) version of INTLOAD is used)

- Ensure the library containing the IC31PLnn load module (if coded) is in the STEPLIB concatenation for the Intercomm Execution JCL

- Execute Intercomm: if POOLSTRT is included and it issues message MI071R at startup, review the above installation steps and try again.

WARNING: if POOLSTRT is in the Intercomm linkedit but there is no 24-Amode pools module in the linkedit, then POOLSTRT will first try to load it (issues existing message MI070R). If an ICPOOLxx 24-Amode pools module is to be loaded, procede as before installing the enhancements (see Operating Reference Manual). Otherwise, to avoid message MI070R, include in the Intercomm linkedit a 24-Amode pools module with at least 1 ICOMPOOL macro coded for at least 1 pool block (COREACCT macro may be omitted).

MACRO CHANGES

The following changes to macros have been made:

COREACCT - internal change - no longer copies INTGLOBE (not needed).

ICOMPOOL - internal change - adds 2 fullwords to POOLACCT Csect entry for each defined pool size - used for header checking in MANAGER, TRAP, TRAP31, and POOLDUMP, and for Pool Detail statistics by RMTRACE.

SPALIST - IC31PL=nn/NO parameter added to define 31-Amode pools table IC31PLnn suffix: code as 1 or 2 byte alphanumeric value. The default is NO indicating the 31-Amode pools table is not to be loaded at startup. (SEXP31SF)

STORAGE - if LOC=ANY (requesting 31-Amode storage) is coded, the request will be satisfied from the 31-Amode pools if available (POOLS=NO not forced), except if POOLS=NO, or the SP parameter, or BOUND=PAGE is coded to force a GETMAIN. If BOUND=PAGE is coded, it is not necessary to also code POOLS=NO (internally forced). If the ADDR parameter is coded, and LOC=ANY is coded, then ADDR may point to a 31-bit address (address of acquired 31-Amode pool or GETMAIN storage is always stored at the ADDR location in 31-Amode by MANAGER). In all other cases (ADDR intended to point to 24-Amode storage), ensure a 24-bit (high-order byte is zero) pointer is passed (a LA instruction will clear the high-order byte if the object is a label in 24-Amode storage, but will not clear it if 'LA Rx,0(Rx)' is coded in 31-Amode execution). If the LIST parameter is used, then it must always point to 24-Amode storage (if LIST omitted, then RENT=NO is forced). The SPA parameter may be used even if executing in 31-Amode.

STORFREE - user must ensure high-order byte of the address passed for the ADDR parameter is zero if trying to free 24-Amode storage. The SPA parameter may be used even if executing in 31-Amode.

PASS - the SPAEXT parameter may be used even if executing in 31-Amode.

CATCH - the SPAEXT parameter may be used even if executing in 31-Amode.

SUBLINK - the first 18 words of the save area requested via an internal STORAGE is cleared by MANAGER. Up to 32760 bytes of storage may be requested.

RTNLINK - value for LEN parameter may be up to 32760 bytes for storage to be freed.

ICOMGEN - DYNPOOL parameter has P31 option added (to YES/NO) for ICOMLINK.

ICOMLINK - DYNPOOL specifies whether or not 31-Amode and/or 24-Amode Intercomm pools modules are to be loaded at startup and an INCLUDE for POOLSTRT is to be generated. Code P31 if only 31-Amode pools are to be loaded (causes an INCLUDE for 24-Amode NEWPOOLS to be generated). Code YES if 24-Amode, and optionally 31-Amode, pools modules are to be loaded (no INCLUDE for NEWPOOLS is generated). The default is NO (NEWPOOLS included, no 31-Amode pools to be loaded, an INCLUDE for POOLSTRT is not generated).

NEW/CHANGED MESSAGES

MI070R ENTER 2-DIGIT SUFFIX FOR ICOMPOOL LOAD MODULE

POOLSTRT

No 24-Amode pools module in the Intercomm linkedit, therefore dynamically loaded 24-Amode Core Pools are required: reply with a 2-digit decimal number nn, where nn is the suffix of the core pools load module ICPOOLnn to be used during Intercomm execution.

MI071R {LOAD} FAILED FOR ICOMPOOL CSECT '{ICPOOLnn}',
{BLDL} {IC31PLnn}
REPLY 'RETRY','CANC', OR 'CONT'

POOLSTRT

nn is the reply to MI070R (if ICPOOLnn), or value coded for IC31PL parameter on the SPALIST macro (if IC31PLnn). Before attempting to load the core pools module, a BLDL was issued to find it in a STEPLIB library which failed, or LOAD of the module returned an address in the wrong Amode for the Csect name. Reply one of the following:

RETRY - if BLDL and ICPOOLnn: ensure it is on STEPLIB, if LOAD and ICPOOLnn: relink it as RMODE=24, AMODE=24 (message MI070R will be reissued); if BLDL and IC31PLnn: ensure it is on STEPLIB, if LOAD and IC31PLnn: relink it as RMODE=ANY, AMODE=31 (before replying).

CANC - Intercomm returns to MVS with step condition code 16.

CONT - The run continues without the pools module - all STORAGE macro requests for that pool type (24- or 31-Amode) will cause GETMAINs.

MI072I INTERCOMM CONTINUING WITHOUT {ICPOOLnn}
{IC31PLnn}

POOLSTRT

Reply to MI071R was CONT.

NEW/CHANGED MESSAGES, con't.

RM002A STORAGE REQUEST LENGTH OF ZERO. PURGING THREAD.

MANAGER 0C1 via ISK

STORAGE macro issued with length of zero. For debugging: get issuing program's address (R14) from the inline save area of the RSMGMNT Csect labeled with the literal RMSAVE, which is the fourth area snapped in an indicative dump. Caller's registers are saved in this area in the order 14-12. This area does not contain chain words as in a standard save area. If register 14 points into PMILINK2, then R8 in RMSAVE is the address of the call to PMILINK2 via a LINKAGE macro. If register 14 points into PMISUBL2, then the fourth word in a static save area at offset X'88' in the SPAEXT is the address of the call to PMISUBL2 via a SUBLINK macro. If register 14 points into SWMODE, then the caller is a program loaded above the 16M line: R13 in the SPIE SAVE AREA is the address of SWMODE's save area if the first byte (of the first word) has a X'31'- if so, then chain back one save area to find the caller's 31-bit return address in the fourth word of the higher save area. However, if R13 does not point to SWMODE's save area, then the caller is the issuer (via SWMODE) of a LINKAGE or a SUBLINK macro, and the remarks above (on R8 or SPAEXT) apply. In a full region snap, use R12 (RSMGMNT base register) in the SPIE SAVE AREA to find the MANAGER module, and then find the RMSAVE literal and proceed as described above.

RM003A STORFREE LENGTH IS ZERO. PURGING THREAD.

MANAGER 0C1 via ISK

STORFREE macro issued with a zero value for the LEN parameter. Register 9 contains the address of the block being freed. If register 14 in the RMSAVE area (see RM002A) points into PMIRTLR, then register 13 in the SPIE SAVE AREA points to the save area of the routine that called the issuer of a RTNLINK macro, and R9 points to the issuer's save area. If R14 points into SWMODE, then the remarks on R13 in RM002A apply, except when the caller is the issuer of a RTNLINK macro, in which case, the return address is in R3 in RMSAVE.

RM005A STORFREE LENGTH GREATER THAN LENGTH OF AREA TO BE FREED. PURGING THREAD.

MANAGER 0C1 via ISK

STORFREE macro issued with an invalid length for a 24-bit or 31-bit pools address. In addition to the comparison of the passed length with the pool header count of remaining bytes in use in the block, a comparison is also made with the resource length field in the RCB. If the passed length is greater than either of these values, this message is issued. Or, the address of the area to be freed is within the range of the pools area, but the passed length to be freed is greater than that of the largest pool in the corresponding area (may indicate an invalid passed address for the Amode of the pools area - 31-bit address when it should be 24-bits (high-order byte not zeroed), or vice versa). In the SPIE SAVE AREA, the passed length is in R8, the address of the area to be freed in R9, the RCB address in R10, and the pool header address in R4 (except if the free length is greater than the largest pool size for the Amode of the passed address). Remarks under RM002A and RM003A apply for finding the STORFREE caller.

NEW/CHANGED MESSAGES, con't.

RM012A {STORAGE } FOUND INVALID POOL HEADER AT xxxxxxxx.
{STORFREE}
{AVRAGING}

MANAGER Snap 91, 92 or 93

If **STORAGE** (see snap 92), then the 8-byte pool block header at the address xxxxxxxx (which was pointed to by ICOMCHN (free chain anchor) for that pool area) has been overlaid: either the next free block pointer (first 4 bytes of header) is invalid, or the second 4 header bytes contain an invalid BLOKCHN or BLOKLEN value for that pool area (see BLOKCON Dsect in copy member RMDSECTS). The free chain pointer in ICOMCHN is zeroed and, after this message and snap 92 are issued, the STORAGE request is obtained from the MVS subpools. If **STORFREE** (see snap 93), then the pool header at address xxxxxxxx for the pool block being freed has been overlaid: either the first 2 header bytes (BLOKCNT) is zero or higher than BLOKLEN, or BLOKCHN or BLOKLEN is invalid (see STORAGE description above). If only BLOKCNT is overlaid, then after this message and snap 93 are issued, BLOKCNT is reset to the value in BLOKLEN and STORFREE processing continues. Otherwise, only the associated RCB is freed or adjusted, depending on the length passed to STORFREE. If **AVRAGING** (see snap 91), then the free chain pointer at address xxxxxxxx is invalid because it is not within the address range of the pool size for which the free blocks are being counted (for pool use detail statistics reporting). After snap 91 and this message are produced, further core accounting statistics gathering and reporting is suppressed for the remainder of the run. In all cases, a Thread Dump is also produced to aid in debugging (look for owner of preceeding pool block).

RM021I STORAGE REQUEST FAILED. LEN=yyyyyy, FROM=xxxxxxxx

MANAGER

yyyyyy is the number of bytes requested; xxxxxxxx is the return address to the module issuing the STORAGE request. This message indicates a low-storage condition if the byte count is reasonable, or a programming error if it is very high. If RENT=NO was not coded or forced for the STORAGE macro, Resource Management will wait and retry to obtain the storage up to the number of times coded on the SPALIST macro for the NTIMS parameter (default is 7). This message is issued at the beginning of the retry sequence; therefore, it does not necessarily mean that the storage request was not satisfied. If storage still can not be obtained or if RENT=NO was coded, MANAGER will return to the caller with a non-zero return code in register 15. Note that PMILINK2, the module invoked by a LINKAGE macro, will program check if it receives a non-zero return code. See also message RM023I.

NEW/CHANGED MESSAGES, con't

RM022A OWNER OF POOL AT xxxxxxxx HAS DESTROYED POOL HEADER AT yyyyyyyy

MANAGER Snap 94

RMINTEG processing has been activated to validate the header of the next higher block to that of a block being STORFREE'd in the 24-Amode pools, and has found the neighbor header to be invalid. xxxxxxxx is the address of the header of the block being freed, and yyyyyyyy is the address of the overlaid header (or of the POOLEND Csect if the last block in the ICOMPOOL Csect is being freed). See the associated Thread Resource Dump (on SMLOG data set) to determine the owner of the block being freed. Then look at that storage and the overlaid header to determine the cause of the overlay by the processing thread. See also the description of snap 94 for further details to aid in debugging this problem.

RM027I NO COREACCT OR POOLS CSECTS. BYPASSING DETAIL CORE USE STATISTICS

MANAGER

The Csects generated by the COREACCT and/or ICOMPOOL macros cannot be found in the Intercomm linkedit and neither 24-Amode nor 31-Amode Intercomm pools modules were loaded. Size request distribution and pool use detail statistics will not be produced for this Intercomm execution. See Operating Reference Manual.

RM029A ADDR PARAMETER FOR STORAGE MACRO NOT 24-AMODE.

MANAGER 0C1 via ISK

STORAGE macro issued without the LOC=ANY parameter (for 31-Amode storage), but the 'where-to-store' address for 24-Amode storage (when obtained) is a 31-bit address, which is not permitted. That is, the pointer value coded for the ADDR parameter on the STORAGE macro is a 31-Amode address, which is only permitted if LOC=ANY is coded. Remarks under RM002A and RM003A apply on how to find the macro issuer. In the SPIE SAVE AREA, register 9 contains the ADDR parameter value passed to STORAGE. Omit or correct the ADDR parameter coding.

NEW/CHANGED ISK CODES

MANAGER **0,0**

Cause: STORAGE requested with a 31-bit parameter list address, which is not allowed.

Debugging Hints: R1 in the RMSAVE area (see message RM002A) contains the parameter list address. Correct the LIST parameter for the STORAGE macro (see message RM002A on how to find STORAGE issuer).

MANAGER **0,5**

Cause: STORAGE requested with an ADDR parameter specifying a main storage location which is not fullword aligned.

Debugging Hints: See message RM025A. R9 in the SPIE SAVE AREA contains the value passed for the ADDR parameter.

MANAGER **1,5**

Cause: STORAGE requested with a 31-bit ADDR parameter, but LOC=ANY (for 31-Amode storage) was not coded.

Debugging Hints: See message RM029A.

MANAGER **1,9**

Cause: STORAGE requested with an SP parameter (forcing a GETMAIN) for an MVS protected subpool, but MRSVC support has not been installed to permit addressing the obtained storage by the requestor. Correct SP number parameter (must be lower than 128).

Debugging Hints: R2 in the SPIE SAVE AREA contains the SP value in its low-order byte from the 3-word parameter list passed to STORAGE via R1. See message RM002A on how to find the STORAGE issuer.

MANAGER	5,0	deleted
MANAGER	6,0	deleted
MANAGER	7,0	deleted

NEW SNAPS

91 MANAGER RM012A

Issued, by the averaging routine for pool use statistics gathering, if the header of a free 24-Amode or 31-Amode pool block is invalid as described for message RM012A.

Registers snapped include:

R2 = ICOMCHN pointer to a pair of addresses for the pool area with the invalid header: the first address is that of the header of the first pool block in that pool area, the second address is that of the header of the first free block in that pool area.
R5 = address of the invalid free block header (see BLOKCON Dsect in RMDSECTS).
R7 = address of the header of the first block in the next pool area (for next higher size), or is the address of POOLEND (end-of-pools delimeter - invalid header is in last pool area).

Areas snapped:

RM012A message text.
ICOMCHN Csect: first address is that of the ICOMPOOL Csect, which is followed by pairs of addresses (as described above for R2) for each pool size requested by ICOMPOOL macros for this pools table.
ICOMPOOL Csect: all pool areas are snapped (header overlay came from the owner of a block preceding the invalid pool header). Look at pool areas preceding the address in R5 to determine the type of overlay, and use the associated Thread Resource Dump called by RSMGMNT (on SMLOG data set) to determine the owner of the pool block where the overlay started.

92 MANAGER RM012A

Issued only if the header of the first free 24-Amode or 31-Amode pool block, that would satisfy a STORAGE request, is invalid as described for message RM012A.

Registers snapped include:

R2 = ICOMCHN pointer to a pair of addresses for the pool area with the invalid header: the first address is that of the first pool block satisfying the requested length, the second address was that of the first free block in that pool area (zeroed before snap).
R4 = address of first free block header (see BLOKCON Dsect in RMDSECTS).
R8 = length requested via STORAGE macro (in bytes).

Areas snapped:

RM012A message text.
RMSAVE REGS 14 TO 12 (STORAGE macro issuer's registers).
ICOMCHN Csect: first address is that of the ICOMPOOL Csect, which is followed by pairs of addresses (as described above for R2) for each pool size requested by ICOMPOOL macros for this pools table.
All pool blocks for size associated with requested length (if R4 points to the header of the first block in the pool area (first address pointed to by R2), then all pool blocks from the beginning of the ICOMPOOL Csect through this pool area are snapped (header overlay came from the owner of a block in preceding pool(s) area). Look at pool areas preceding the address in R4 to determine the type of overlay, and use the associated Thread Resource Dump called by RSMGMNT (on SMLOG data set) to determine the owner of the pool block where the overlay started.

NEW SNAPS con't.

93 MANAGER RM012A

Issued only if the header of the 24-Amode or 31-Amode pool block, for which a STORFREE has been requested, is invalid as described for message RM012A.

Registers snapped include:

R4 = address of invalid pool block header (see BLOKCON Dsect in RMDSECTS).
R8 = length requested to free via STORFREE macro (in bytes).
R9 = address of area within block (starting at R4 + 8) to be freed.
R10 = address of found RCB for storage to be freed (see RCB Dsect in RMDSECTS).

Areas snapped:

RM012A message text.
RMSAVE REGS 14 TO 12 (STORFREE macro issuer's registers).
ICOMCHN Csect (see description in snap 92).
All pool blocks for size associated with requested length (if R4 points to the header of the first block in the pool area, then all pool blocks from the beginning of the ICOMPOOL Csect through this pool area are snapped (header overlay came from the owner of a block in preceding pool(s)). Look at pool areas preceding the address in R4 to determine the type of overlay, and use the associated Thread Resource Dump called by RSMGMNT (on SMLOG data set) to determine the owner of the pool block where the overlay started.

NEW SNAPS con't.

94 MANAGER RM022A

Issued by RMINTEG processing. When activated, RMINTEG is called on a STORFREE of a 24-Amode pool block with a valid header to test if the header of the next (neighbor) pool block (or POOLEND Csect if the last pool block being freed) is valid, as described by message RM022A.

Registers snapped include:

R2 = ICOMCHN pointer to a pair of addresses for the pool area with the invalid header: the first address is that of the header of the first pool block in that pool area, the second address is that of the header of the first free block in that pool area.
R4 = address of header of pool block being STORFREE'd.
R5 = if not zero, and the same as R7, then the overlaid header is for a pool block on the free chain (and will be made the end of the free chain); if not zero, and not equal R7, then it is the address of an overlaid header on the free chain: if this is the first snap (see snap 91) indicating this header, then there is an extensive overlay (if R5 address higher than R7), or more than one header overlay (also use the Thread Resource Dump called by RMINTEG to help debug this problem).
R6 = if not zero, then only the second word of the neighbor header was found invalid.
R7 = address of overlaid header (or of POOLEND Csect).
R10 = if not zero, is the RCB address for the neighbor pool block, which is in use.

Areas snapped:

RM022A message text.
RMSAVE REGS 14 TO 12 (STORFREE macro issuer's registers).
ICOMCHN Csect: first address is that of the ICOMPOOL Csect, which is followed by pairs of addresses (as described above for R2) for each pool size requested by ICOMPOOL macros for this pools table.
ICOMPOOL Csect: all pool areas are snapped (header overlay came from the owner of the block preceding the invalid pool header). Look at the pool area whose 8-byte header address is in R4 preceding the address in R7 to determine the type of overlay.
POOLEND Csect: 8-byte area consisting of a fullword of X'FF', followed by a fullword of X'00'.

NOTE: if this is not the first snap for this overlaid header, and the overlay is similar to that in the previous snap (which may be a snap 91, 92, 93, or 94), then either deactivate RMINTEG processing via the STOP system command, or deactivate the offending subsystem (if owner of area being STORFREE'd is not thread 0) via the DELY command if the subsystem is loadable or by setting SCHED to NO in screen 2 of the FTUN command response (and change RLSE to SSUP in the verb position - see System Control Commands). If the overlay is due to a COBOL subsystem's DWS area, or a PL/1 subsystem's ISA area, being too small, use the SPAC command to dynamically correct the size before resuming processing by that subsystem. If the overlay is due to a problem in thread 0 (system module) processing, and occurs more than once, then cancel the system with a dump and contact Intercomm system support.

INTERNALS CHANGES

- &RM and &RMPOOLS globals deleted from INTGLOBE and SETGLOBE.
- POOLACCT Csect increased by 8 bytes per ICOMPOOL macro (for header overlay testing).
- RMDSECTS (Storage Management Areas Dsects) expanded for 31-Amode storage request and pools statistics. 24-Amode statistics labels changed.
- The 2 dispatched storage management routines appearing on the IJKTRACE (Timer WQE's) printout and formerly listed only as 'RSMGMNT+displacement' have been changed: the statistics averaging routine for the COREACCT distribution statistics is in MANAGER with a special Entry Point of **AVRAGING** (dispatch frequency controlled by SPALIST macro, RMSTIM parameter); while the printing of the CORE USE STATISTICS is done by the self-dispatching independent subroutine (and Csect) **RMTRACE** (dispatch frequency controlled by SPALIST macro, TRACETM parameter).
- The 2-byte character field in the SPAEXT containing the user-coded IC31PLxx suffix is labeled SEXP31SF (default is 'NO' = no 31-Amode pools to be used).
- SPAEXT 4-byte field labels added for 31-Amode Pools are:

NAME	DESCRIPTION	SPAEXT OFFSET
SEXP31CA	Address of 31-Amode COREACCT Csect	+01C4
SEXP31NX	Address of 31-Amode ICOMINX Csect	+0258
SEXP31CH	Address of 31-Amode ICOMCHN Csect	+025C
SEXP31PL	Address of 31-Amode ICOMPOOL Csect	+0260
SEXP31PE	Address of 31-Amode POOLEND Csect	+0264
SEXP31HL	Address of 31-Amode HILIM Entry (max pool size)	+0268
SEXP31PA	Address of 31-Amode POOLACCT Csect	+026C
SEXP31PR	Address of 31-Amode POOLREGS Entry	+0270
SEXP31PC	Address of 31-Amode POOLCONS Csect	+0274